SIMULTANEOUS RANGE-VELOCITY PROCESSING AND SNR ANALYSIS OF AFIT'S RANDOM NOISE RADAR

THESIS

T. Joel Thorson, Captain, USAF

AFIT/GE/ENG/12-40

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT/GE/ENG/12-40

SIMULTANEOUS RANGE-VELOCITY PROCESSING AND SNR ANALYSIS OF AFIT'S RANDOM NOISE RADAR

THESIS

Presented to the Faculty

Department of Electrical and Computer Engineering

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

in Partial Fulfillment of the Requirements for the

Degree of Master of Science in Electrical Engineering

T. Joel Thorson, B.S.E.E., M.A. Organizational Management

Captain, USAF

March 2012

AFIT/GE/ENG/12-40

SIMULTANEOUS RANGE-VELOCITY PROCESSING AND SNR ANALYSIS OF AFIT'S RANDOM NOISE RADAR

T. Joel Thorson, B.S.E.E., M.A. Organizational Management
Captain, USAF

Approved:

Geoffrey A. Akers [illegible]

Geoffrey A. Akers, PhD (Chairman)

21 Feb 2012
Date

[illegible]

Peter J. Collins, PhD (Member)

21 FEB 2012
Date

[illegible]

Michael A. Temple, PhD (Member)

21 Feb 2012
Date

AFIT/GE/ENG/12-40

Abstract

A United States Air Force autonomous vehicle sensor must be compatible with other electromagnetic devices, accurately detect and track obstacles, and be persistent in all types of weather and harsh environments. Additionally, the sensor must be low-power due to limited system power availability and offer a low probability of detection to maintain a covert posture. The Air Force Institute of Technology random noise radar (RNR) has been shown to possess all of these favorable characteristics, but has not been configured for an autonomous vehicle collision avoidance application. Two primary research objectives include advancing the RNR signal processing algorithm and modeling capability, with an overarching goal of performing collision avoidance on an autonomous vehicle. These objectives are addressed using analytical, simulated, and measured results.

The current RNR signal processing algorithm does not perform simultaneous range and velocity (range-velocity) processing of the receive signal. The continuous, bandlimited, thermal noise transmit signal has a high fractional bandwidth that makes classical Doppler processing impractical. A previous research effort implemented a 2D time domain processing algorithm, but the lengthy signals required for 2D processing made real-time range-velocity processing out of reach. Additionally, the random noise signal eliminates *a priori* reference signal generation, adding to the real-time processing requirements. The first research objective is aimed at reducing the memory required for 2D time domain processing in order to distribute the processing algorithm across hundreds of processors on a GPU. Distributed processing reduces the overall 2D processing time and the feasibility of a near real-time implementation is studied.

The second research objective consists of improving a Simulink® model of the AFIT RNR. Each component of the AFIT RNR, as well as the target environment, is modeled and compared to measured results. A robust model enables efficient signal-to-noise ratio (SNR) analysis of the RNR at all points within the radar system. A thorough SNR analysis is foundational to determining the RNR detection capabilities of the RNR and will benefit future development for collision avoidance applications.

Acknowledgements

First, and above all else, I give thanks and praise to the one from whom all blessings are given. Thank you, God, for continually giving generously to me and my family. Thank you for giving your son, Jesus Christ, who paid the price that you knew I could not pay. To my beautiful bride...thank you for your encouragement and uplifting words that motivate me from the core. Thanks for your commitment to an eternal cause and for your investment in my life and in the life of our kids. To the four greatest kids in the world...thank you for putting up with me while I was gone early in the mornings and late at night. Thank you for welcoming me home each day with big hugs and happy smiles. Every day I am proud of each of you. To my parents...thank you for instilling in me a desire to learn and teaching me how to be a man of character. A heartfelt thanks goes to my advisor, whose counsel extended beyond the research. Thank you, Lt Col Akers, for modeling your guidance after the Wonderful Counselor, and thank you for the time you committed to me and to all the LORE students at AFIT. Thanks also to the top-notch faculty at AFIT who take people like me who can't even spell "radar" and turn us into functioning engineers. Finally, I would like to give a shout-out to the guys in the RAIL. The life conversations, sports talk, jokes, study sessions, and brainstorming were vitally important to my personal and academic growth. The list of people that have invested in me during this research effort is amazingly long. I'm humbled by your investment and will deeply value each of you always.

T. Joel Thorson

Table of Contents

Page

List of Figures

Figure Page

Figure Page

List of Tables

SIMULTANEOUS RANGE-VELOCITY PROCESSING AND SNR ANALYSIS OF AFIT'S RANDOM NOISE RADAR

I. Introduction

1.1 Problem Description

Autonomous vehicles must be outfitted with sensors to provide the obstacle identification and warning information required for collision avoidance. These sensors must be able to precisely estimate the location and relative velocity of nearby objects to determine if course correction is required to avoid unwanted collision. Additionally, each sensor must be able to operate in the presence of other sensors and vehicle systems. Electromagnetic compatibility between the sensors and vehicle systems is critical to reliable vehicle operations. To be desired by design engineers, the sensor must be small, light-weight, low-power, electromagnetically compatible, and accurate. The Air Force Institute of Technology (AFIT) random noise radar (RNR) offers a low peak power, accurate and compatible solution. However, the AFIT RNR size, weight and signal processing algorithm are not practically configured for an autonomous vehicle collision avoidance application.

Research on the AFIT RNR began in 2007 with Schmitt and focused primarily on through-the-wall (TTW) imaging using a network of multistatic radar nodes [27, 33, 34]. The research continued with enhancements to the network of noise radars, including the development of a basic software model [30]. Recently, the research was extended to include a thorough characterization of the AFIT RNR's ability to process range and velocity information simultaneously [18]. Although real-time simultaneous processing of range and velocity has been proven using a RNR with a hardware defined correlation receiver [17, 25, 26], it has not been demonstrated using the direct conversion receiver (DCR) implemented in the AFIT RNR.

The inherent qualities of the AFIT RNR, including its low peak power, electromagnetic compatibility, and accurate waveform, point to a promising future as a collision avoidance sensor. A problem arises, however, in that the size, weight, and signal processing algorithm used in the AFIT RNR does not lend the system to a practical collision avoidance application. The form factor and weight must be reduced, the algorithm must interface with the control system of the vehicle to ensure timely course corrections, and the system must be able to simultaneously process range and velocity information in near-real time in order to take full advantage of the noise radar capabilities as a collision avoidance sensor.

1.2 Research Motivation

Radar, electro-optical (EO), and infrared (IR) systems are the three primary capabilities used by the United States Air Force (USAF) for remote sensing. EO and IR sensors are passive systems that use external stimuli to illuminate an environment in order to glean information from that environment. Unlike EO and IR systems, radar is an active sensor that emits an electromagnetic (EM) wave and receives the wave's return echo to gather information about an environment. Radar has a distinct advantage over passive systems in that it can search, track, and/or image a desired scene in all weather conditions, both day and night. While EO sensors can operate only in the presence of a light stimulus, and IR sensors can be significantly impacted by weather and temperature conditions, radar systems do not have such limiting constraints. Radar, however, has a number of distinct disadvantages. Classical pulse radars require high peak power to achieve significant detection range, resulting in limited electromagnetic compatibility with other collocated systems. Additionally, the periodicity of the EM wave along with its high peak power make classical pulse radar easily detectable by uncooperative observers, complicating covert operations. With its low peak power and continuous, random signal, the random noise radar can eliminate the primary limitations of classical radar by offering electromagnetic compatibility and a covert posture while maintaining the ability to gather target information in any environment.

Collision avoidance sensors are a necessary evil for autonomous vehicle designers. An autonomous vehicle for any practical application, particularly in the USAF, is not designed solely to move from point A to point B. It is designed to carry a payload and perform a mission. The fact that a USAF vehicle is autonomous ensures that the mission can be carried out without placing forces in harm's way. Autonomous vehicle designers face an engineering problem. They must design the system to travel according to the mission plan while avoiding obstacles, but they must do it without sacrificing the capabilities of the primary mission payload. With a limited amount of size, weight and power available to the autonomous vehicle, the requirements of the collision avoidance sensors often conflict with the requirements of the mission payload. Additionally, in the case of a remote sensing payload, the collision avoidance sensors must not interfere with the payload sensors. They must be electromagnetically compatible.

Researchers have shown that a random noise radar can be used for automobile collision avoidance [21, 23]. Lukin [23], who demonstrated an experimental collision avoidance system in an urban environment, concluded that noise radar technology offers the most suitable solution for automobile collision avoidance systems. He based his conclusion on the fact that noise radars require low power, can be constructed in a small package, can operate in the presence of other radar equipped vehicles, are immune to electromagnetic interference, are low cost and light weight, and offer excellent resolution.

The AFIT RNR has been proven as a low-power, accurate, and covert sensor [27, 30, 33]. Configuring the AFIT RNR for autonomous vehicle collision avoidance could give the vehicle designer a capable solution to the tradeoff between obstacle avoidance and mission payload sensors. The AFIT RNR has the potential to provide accurate and compatible sensor information for autonomous vehicle collision avoidance without placing a significant burden on the size, weight, or power available to the vehicle systems.

1.3 Research Goals

In order to be used for collision avoidance on an autonomous vehicle, the form factor and weight of the AFIT RNR hardware must be reduced. Additionally, the signal processing algorithm must be modified to support collision avoidance. In a parallel research effort, Ludwig is focused on exploiting antenna and other hardware modifications suitable for autonomous vehicle applications [20]. The research effort presented here, however, will focus on optimizing the signal processing algorithm for collision avoidance. The primary objectives of this research effort include:

1. Minimize the simultaneous range and velocity processing time in the AFIT RNR through parallelization.

2. Characterize the AFIT RNR in terms of the signal-to-noise ratio (SNR).

The secondary objectives include:

1. Configure the signal processing algorithm for parallel processing on a field-programmable gate array (FPGA) or graphics processing unit (GPU).

2. Update the AFIT RNR Simulink® model to match measured data.

These objectives will help provide the trade-space analysis for advancing the AFIT RNR toward a practical collision avoidance application.

1.4 Background

The term "random noise radar (RNR)" is also referred to in literature as noise radar technology (NRT), random signal radar (RSR), and simply noise radar. Regardless of the label, random noise radars represent a class of radars that use an electromagnetic wave with the characteristics of thermal noise to illuminate a target within its surrounding environment. The concept of noise as a random waveform is not new, tracing its roots to Christian Huelsmeyer in 1904, who used noise in his radar precursor, the "telemobiloscope" [12], which detected ships at ranges up to 3 km in all weather conditions [13]. Later, in 1957 and 1959, Richard Bourret and B. M. Horton introduced coherency into the noise radar receiver [22]. Horton discovered that a noise waveform could be used to eliminate range and Doppler ambiguities by correlating the return signal with a time delayed replica of the transmitted noise waveform. Although some advances in noise radar technology occurred over the next three decades [10], there was comparatively little research in that time period due to the highly complex components required for correlation signal processing and the lack of efficient noise waveform generators. In the 1990s and 2000s, advances in digital signal processing (DSP) hardware and software made advancement in noise radar technology feasible.

Today, noise radar researchers are finding numerous applications for the capability, including synthetic aperture radar (SAR) imaging [9], inverse SAR (ISAR) imaging [6], through-the-wall (TTW) imaging and surveillance [15, 34], sub-surface detection [43], foliage penetration [42], and even as an automobile collision warning sensor [21, 23]. The fundamental concepts of noise radar are now widely known and recent research is focused on application specific design requirements that will make the RNR a suitable alternative to classical radar systems [36].

1.5 Organization of Thesis

Further discussion of the foundational principles of the AFIT RNR and the theoretical development aimed at achieving the research goals is found in Chapter 2. In Chapter 3, the simulation and experimental methodology used in this research effort is presented. Chapter 4 provides a detailed analysis of the results uncovered during the experimental effort along with a performance and utility assessment. The conclusions are discussed in Chapter 5 and include recommendations for future research.

II. Theory

2.1 Chapter Overview

In this chapter, the principles that make up the foundation of noise radar are presented, specifically in the context of the AFIT RNR. The chapter begins with a tutorial of basic radar concepts and transitions to continuous wave, ultra-wideband noise radar. It continues with a discussion of the AFIT RNR's ability to simultaneously process range and velocity information, summarizing previous research and results. Finally, the chapter outlines the underlying principles supporting the SNR analysis of the AFIT RNR.

2.2 Radar

Before entering into a discussion of current noise radar technology and the AFIT RNR, it is important to understand the basic concepts of radar. The term radar was originally an acronym for radio detection and ranging, but with the development of its extensive capabilities, radar is no longer used for just detection and ranging. Today, it is widely used for velocity estimation, imaging, and many other functions. This section highlights the classical pulse radar, the range and velocity estimation capabilities of radar, and general radar configurations.

2.2.1 Classical Pulse Radar.

In a classical monostatic pulse radar, a short burst of energy is transmitted into the environment followed by an extended listen time. If a target is present in the direction of propagation, the wave reflected from the target is received and compared to the transmitted wave to extract target information. The time between transmit pulses is known as the pulse repetition interval (PRI).

As can be seen in Figure 2.1, the time it takes for the pulse to travel to the target and back to the receiver, ΔT, is known as the two-way transit time. Because the velocity of EM waves in free space is known to be the speed of light ($c \approx 3 \times 10^8$ m/s), the range, R, to the target can be extracted from the two-way transit time, and is given by the relationship

$$R = \frac{c\Delta T}{2}. \tag{2.1}$$

For a stationary target, one pulse can determine the range to the target, although multiple pulses will increase confidence that the return signal is not noise resulting in a false alarm.

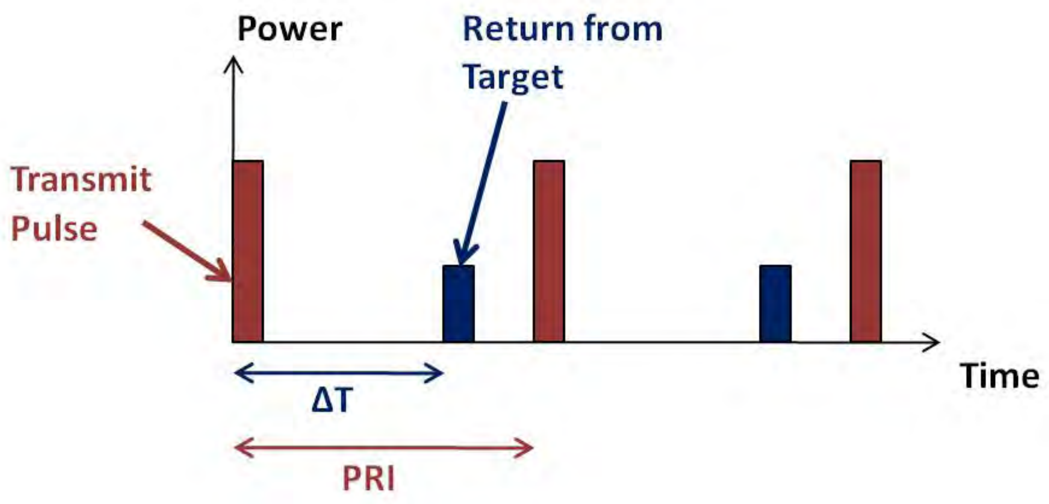


Figure 2.1. The distance to the target can be derived from the two-way transit time ΔT.

Velocity information can be extracted by comparing the range of the target over multiple pulses. Radial velocity, or target velocity in the direction of the antenna line of sight, can be deduced if the measured range varies from pulse to pulse. Similarly, radial velocity can be derived from the difference between the transmit and receive frequencies, called the Doppler frequency, of a single pulse. Because all measurable targets (even supersonic aircraft) travel at speeds much lower than the speed of light, the relationship between radial velocity and Doppler frequency is given by

$$v_r = v\cos\psi \approx \frac{f_d\lambda}{2} \text{ for } |v| \ll c, \tag{2.2}$$

where v is the velocity of the target, v_r is the radial velocity, ψ is is the angle difference between the velocity vector and the radar line of sight, f_d is the Doppler frequency, and λ is the wavelength of the transmitted signal [31]. The Doppler frequency can be extracted only from a phase-coherent system, meaning the phase of the received signal with respect to the transmit signal must be known. Many radars use the same local oscillator (LO) for frequency conversion and for mixing in both the transmitter and receiver in order to maintain phase coherence.

A radar's performance is characterized by the radar range equation (RRE). But to understand the RRE, the SNR must first be defined. The SNR is used by system designers to compare the received signal power, P_r, with the system noise power, P_n. The received signal power, P_r is defined as

$$P_r = \frac{P_t G_t G_r \lambda^2 \sigma}{(4\pi)^3 R^4 L_s}, \tag{2.3}$$

where P_t is the peak transmit power, G_t and G_r represent the transmit and receive antenna gain, σ is the radar cross section (RCS) of the target, and L_s is the system losses. Similarly, the system noise power, P_n, is often defined as

$$P_n = kT_0BF, \tag{2.4}$$

where k is Boltzmann's constant ($1.38e^{-23}$ Joules/K), T_0 is the standard temperature (290 K), B is the instantaneous receiver bandwidth in Hz, and F is the unitless system noise figure. SNR, then, is the ratio between the received signal power and the system noise, defined as

$$SNR = \frac{P_r}{P_n} = \frac{P_t G_t G_r \lambda^2 \sigma}{(4\pi)^3 R^4 k T_0 B F L_s}. \tag{2.5}$$

By solving (2.5) for range, the maximum distance that an object can be detected by the radar is defined as the RRE and given by

$$R_{max} = \sqrt[4]{\frac{P_t G_t G_r \lambda^2 \sigma}{(4\pi)^3 k T_0 B F L_s (\text{SNRmin})}}, \tag{2.6}$$

where SNRmin is the minimum detectable SNR of the system. This is an important relationship for radar designers. If greater detection range is desired, then significant improvements to antenna gain or transmitted power must be realized. The other parameters in (2.6) are often fixed and cannot be altered to improve range.

Pulse radars can be fabricated using analog components, making them producible during the early years of radar. Many capabilities and applications have grown from the pulse radar, but as the technology has developed, so has the exploitation of pulse radar. Even simple electronic support measurement devices can detect pulse radars [27], making covert operations with the traditional pulse radar nearly impossible. Additionally, pulse radars require high peak power to achieve significant detection range. From (2.6), it can be seen that range is determined by the fourth root of power. Doubling the peak power will increase the maximum range of the radar by only 19%. To double the range, power must be multiplied by a factor of 16. Increasing detection range requirements can quickly force changes to the system power supply and can result in electromagnetic interference (EMI) with other devices operating in the same frequency range.

Another significant disadvantage of classical pulse radar is the range and Doppler ambiguity inherent in pulse radar. As a simple example of range ambiguity, consider Figure 2.2. The return echo of the first pulse occurs after the second pulse has been transmitted. The system may not know if this is the return from a distant target reflecting the first pulse, or the return of a near target reflecting pulse two. Many pulse coding techniques have been developed to resolve this type of range and Doppler ambiguity, but it is inherent in pulse systems.

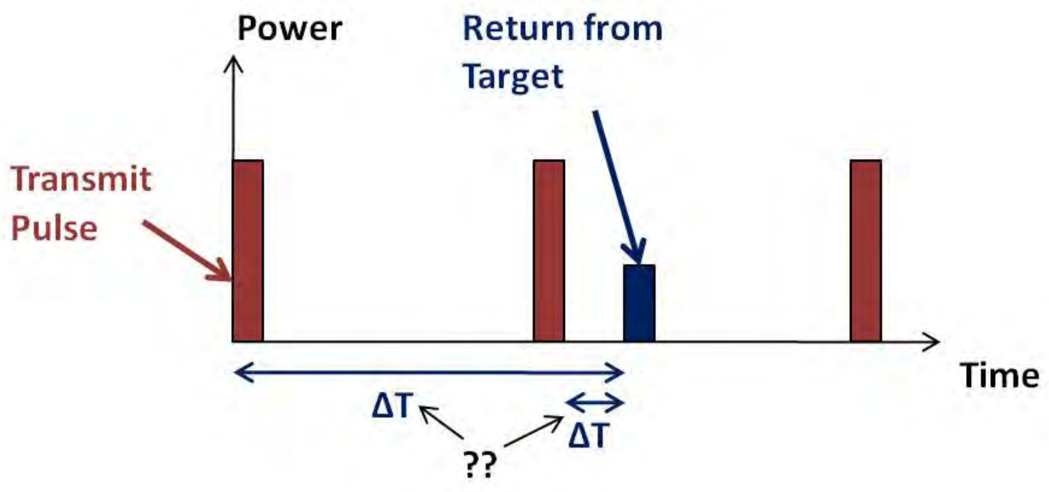


Figure 2.2. It is ambiguous whether this return echo is from the first pulse or the second pulse.

2.2.2 Radar Configurations.

Monostatic and bistatic are the two basic radar configurations [31]. Monostatic configurations have the transmitter and receiver collocated, and in many pulse radar applications, the transmit and receive subsystems share the same antenna. Monostatic systems have the advantage of a single clock source for the transmitter and receiver to ensure coherency. Additionally, the signal processing can be accomplished locally with no need to pass the reference signal to a remote system. The monostatic configuration, however, has a number of disadvantages. First, good separation must exist between the transmit and receive signals. In a pulse radar, this separation can be accomplished with a circulator or switch to protect the sensitive receiver components from the high transmit power. Second, it can be difficult to detect stealth targets, because the target is often designed to avoid backscatter and reflect the signal in directions away from the radar. On the other hand, a bistatic radar can be configured to collect signals that reflect from stealth targets by spatially separating the receive antenna from the transmit antenna. This configuration requires the receiver to have remote access to the reference signal.

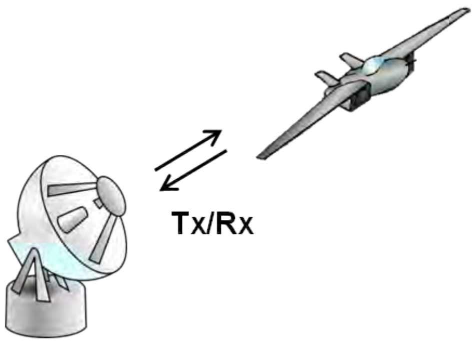
A third radar configuration, multistatic, is simply a combination of monostatic and bistatic configurations. In a multistatic system, there can be any number of combinations of monostatic antennas as well as bistatic antennas to decrease detection losses caused by signal fading, multi path, and target scattering. Multistatic configurations also have the advantage of being able to estimate a target's shape [31]. Figure 2.3 illustrates the various configurations implemented in a radar system.

2.3 Noise Radar

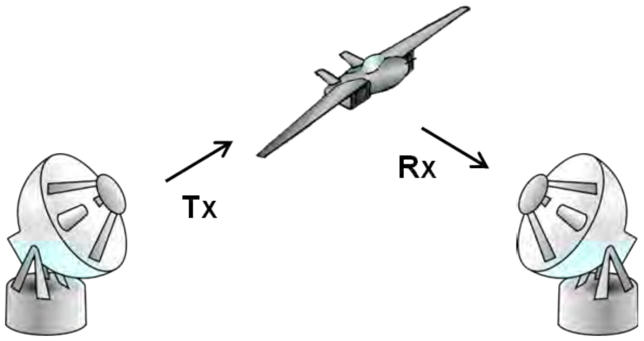
An alternative to classical pulse radar is continuous wave, ultra-wideband noise radar. The distinguishing features of this type of radar are described in this section.

2.3.1 Continuous Wave Radar.

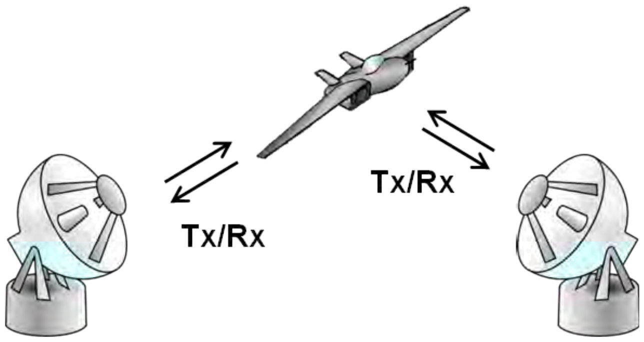
Although the first radars used a continuous waveform, pulse radars quickly grew in popularity. Continuous wave (CW) radars later made a resurgence as an alternative to the high power pulse wave radars [31]. Instead of short bursts of energy, a continually transmitting signal is emitted, resulting in a continuous receive signal. As defined earlier, the monostatic radar configuration uses a single antenna for both transmit and receive. In the case of CW radars, a single antenna is not possible because the radar is continually transmitting and receiving. The transmit antenna, however, can



(a)



(b)



(c)

Figure 2.3. A radar system can be configured as (a) monostatic, (b) bistatic, or (c) multistatic.

be collocated with a separate receive antenna, resulting in a bistatic configuration that functions in a similar manner as the monostatic configuration. Many authors define this bistatic configuration, with a small separation between the transmit and receive antennas, as monostatic or near-monostatic. In this study, it will be defined as a near-monostatic radar configuration.

The major advantage of the continuous waveform is that it can operate with low peak power, as seen in Figure 2.4. Because CW radars operate at such low peak power, they are often undetected by uncooperative receivers. CW radars are an excellent choice for low-power Doppler radars, such as police and other velocity detectors. As can be seen in Figure 2.5, the continuous wave signal requires some form of modulation to measure range. Frequency or phase modulation is often used to vary the characteristics of the wave over time, placing a timing mark on the wave used for reference in determining range [31].

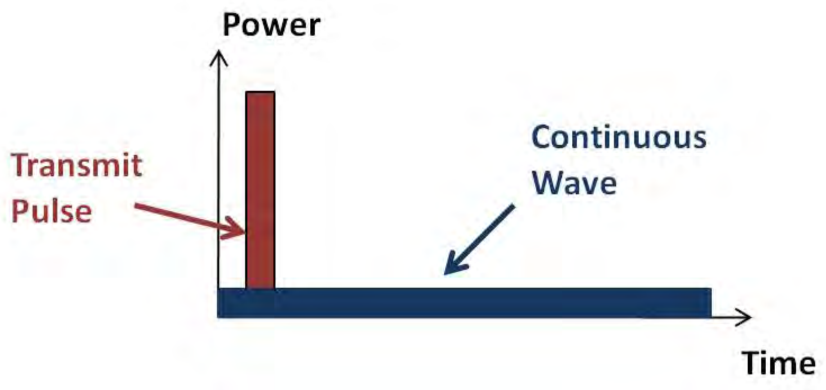


Figure 2.4. Power of a pulsed wave radar compared to an equal energy CW radar signal with similar detection capabilities. (Adopted from [29])

2.3.2 Noise.

Random white Gaussian noise (WGN) may be used to modulate a CW signal in order to extract range information. Noise radars have a number of characteristics that are desired for niche applications. A significant advantage is that its aperiodic, random waveform gives the RNR a low probability of intercept (LPI). Most receivers are designed to filter or suppress noise, so a low-power noise waveform would not

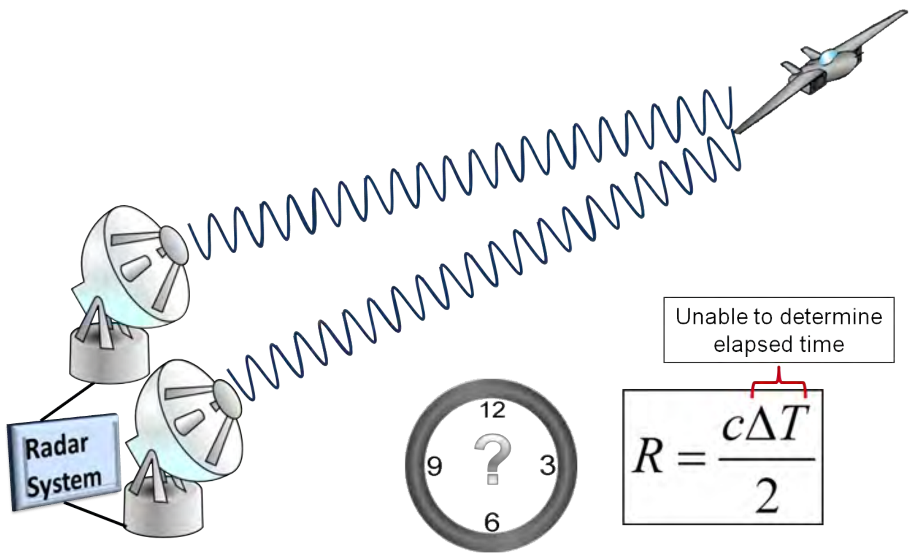


Figure 2.5. Without frequency or phase modulation, a continuous wave signal cannot determine range.

trigger a detect in the uncooperative receiver, meaning the radar can see without being seen. Additionally, RNR systems are resistant to electromagnetic interference, allowing operation in close proximity to other systems in the same frequency band, including other noise radars [36].

A random noise signal, by definition, has unlimited bandwidth and thus it transmits over the entire frequency spectrum. However, due to constraints on hardware and processing, the signal is band-limited by design [27]. Although the transmitted signal can be truly random, it is known to the system which compares the received signal to the transmitted signal. A random signal will not correlate with another random signal unless they are exact replicas, in which case the correlation is a maximum [38], meaning there is no ambiguity in range or velocity, as can be seen in Figure 2.6. More detail on correlation and ambiguity in the RNR will be provided in the following sections.

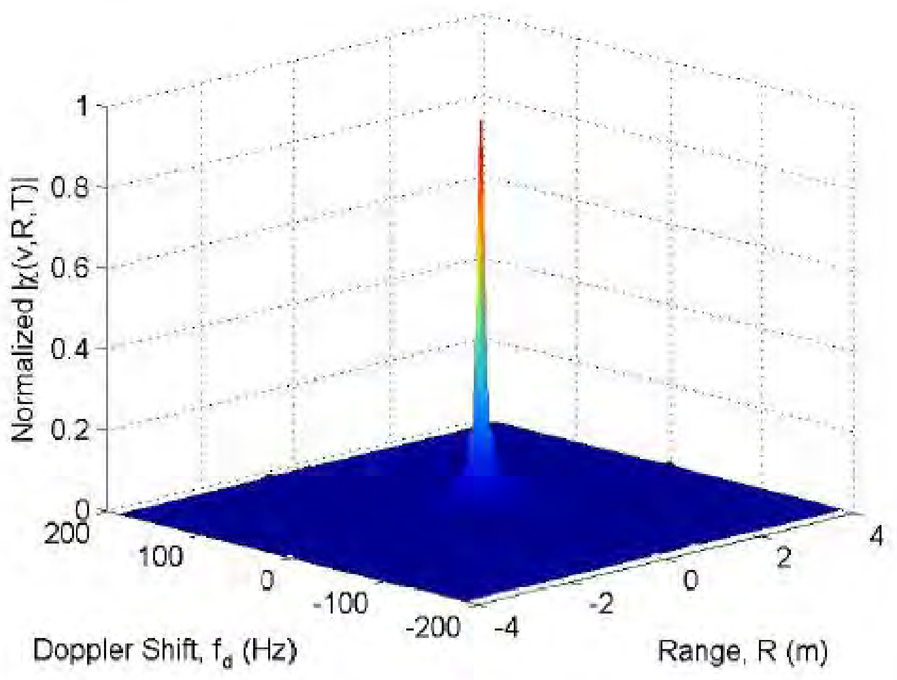


Figure 2.6. The UWB RNR offers theoretical accuracies in both range and Doppler [18].

2.3.3 Ultra-wideband Waveform.

Many radar systems operate at a given frequency or at a very narrow range of frequencies. A number of systems, however, are designed to operate across a wide range of frequencies offering the advantage of improved target range resolution [35]. Instead of being defined by a single carrier frequency, f_c, wideband systems are defined by a range of frequencies from f_{low} to f_{high}. Ultra-wideband (UWB) systems are generally characterized to have over 20-25% fractional bandwidth, $B_{fractional}$, defined as

$$B_{fractional} = \frac{2(f_{high} - f_{low})}{f_{high} + f_{low}}. \tag{2.7}$$

Noise is inherently UWB because random oscillations of electron carriers occur at all measurable frequencies. Wide bandwidth leads to better range resolution. As bandwidth increases, the radar's ability to distinguish two targets increases. This relationship is shown to be

$$\Delta R = \frac{c}{2B}, \tag{2.8}$$

where ΔR is the minimum resolvable target spatial separation. From (2.8), it can be seen that large bandwidth will lead to fine range resolution, making range estimates highly accurate [13].

2.3.4 Noise Waveform.

For many noise radars, the transmit waveform, $s(t)$, is the band-limited output of a thermal noise generator and is statistically modeled as band-limited white Gaussian noise (WGN). The noise amplitude has a Gaussian probability density function with a mean of zero. The power spectral density is uniform, or white, is distributed evenly across all frequencies, and is wide sense stationary (WSS) [11].

In the case of a narrowband radar, the receive signal is a delayed and frequency shifted version of the transmit signal given by

$$s_R(t) = As(t-\tau)e^{j2\pi f_d t + j\phi}, \tag{2.9}$$

where $\tau = 2R/c$ represents the delay caused by the distance between the radar and the target. The general form of the receive signal scales the transmit signal in amplitude by A, and the phase shift given by $e^{j2\pi f_d t + j\phi}$ results from the relative motion between the target and the radar. However, for a wideband radar, the receive signal is shown to be [40]

$$s_R(t) = As(\alpha(t-\tau)), \tag{2.10}$$

where α is due to the relative motion between the target and the radar is modeled by a time scale defined as

$$\alpha = \frac{c-v}{c+v}. \tag{2.11}$$

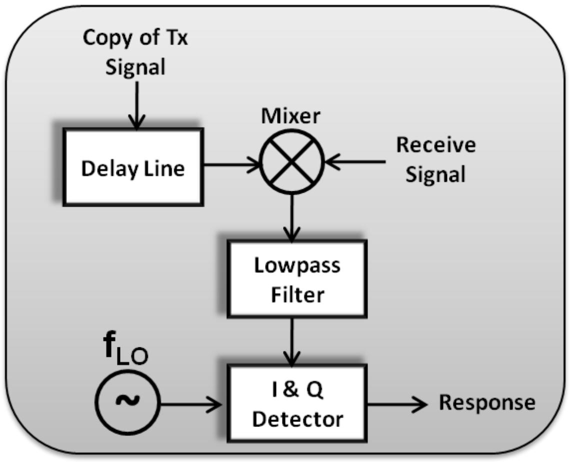
For simplicity, the amplitude scaling factor is ignored, and because $v \ll c$, the receive signal is often represented as

$$\begin{aligned} s_R(t) \propto s(\alpha(t-\tau)) &= s(\alpha t - \alpha\tau) = s\left(\alpha t - \left(\frac{c-v}{c+v}\right)\left(\frac{2R}{c}\right)\right) \\ &= s\left(\alpha t - \frac{2Rc}{c^2+cv} - \frac{2Rv}{c^2+cv}\right) \approx s(\alpha t - \tau) \\ s_R(t) \propto s(\alpha t - \tau). & \end{aligned} \tag{2.12}$$

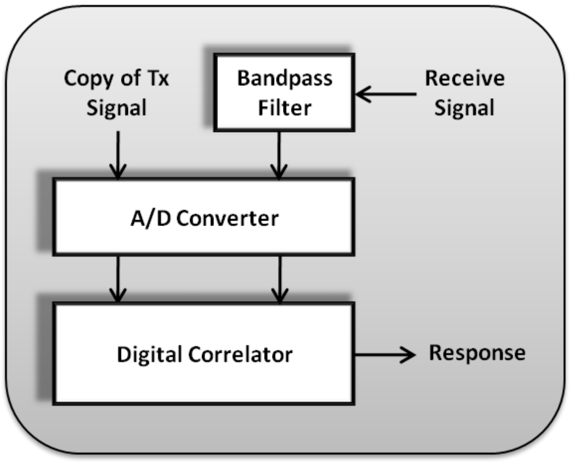
2.4 Signal Processing

Noise radars transmit continuously and randomly over a wide range of frequencies with no carrier frequency. The AFIT RNR only samples the real part of the transmit and receive signals, resulting in recieved signals that are not phase-coherent with the transmitted signals. Narayanan developed a method to inject phase coherency into a noise radar by using heterodyne correlation to compare the received signal with a replica of the transmit signal [25]. The transmit signal is delayed in time and shifted in frequency until a match occurs, making known the range and velocity information of the target.

The AFIT RNR simply filters and amplifies the RF receive signal before passing it to the analog-to-digital converter (ADC), thus implementing a fully-digital correlation receiver called the direct-conversion receiver (DCR). The DCR offers a number of benefits ranging from simplicity of design to minimal additive noise, but the ADC only gathers amplitude samples and the phase information of the transmit signal is not known to the DCR, maintaining the phase-incoherence of the AFIT RNR [27, 30]. A comparison of Narayanan's coherent heterodyne receiver with AFIT's noncoherent digital receiver is illustrated in Figure 2.7. Whether implementing the AFIT DCR, or Narayanan's heterodyne receiver, the foundation of noise radar signal processing is detailed in this section.



(a)



(b)

Figure 2.7. Comparison of (a) the heterodyne receiver as outlined in [25] and (b) AFIT's fully digital correlation receiver architecture [30].

2.4.1 Correlation.

Physical constraints of hardware and limited processing capabilities force CW noise waveforms to be processed in intervals. The time of each interval is known as the integration time, or measurement window, T. In order to satisfy the Nyquist criteria, the signal must be sampled at a rate $f_s = 2B$, leading to a signal of length

$$N = 2BT \tag{2.13}$$

for each interval that has a measurement window, T.

Correlation of two signals is simply a measure of how well the two signals resemble each other. The time delay between the transmit signal and its echo can be estimated by finding the maximum of the cross-correlation between the received signal, $s_R(t)$ and the complex conjugate of the delayed transmit signal, $s(t)$, given by [37]

$$y(\tau) = \int_{t=0}^{T} s_R(t)s^*(t-\tau)dt. \tag{2.14}$$

Essentially, the time delay of the replicated transmit signal is varied until a match occurs. The match will result in a peak of the correlation function at time $t = \tau$, which corresponds to the two-way transit time of the signal used to estimate range as in (2.1). Noise is an aperiodic stochastic process that will only correlate when it is compared to itself. Only a delayed version of the transmit signal will result in a correlation. All other interfering noise is independent and will not correlate, allowing the signal processor to identify a weak echo signal in a noisy environment.

2.4.2 Matched Filtering.

To achieve the fundamental goals of detection, tracking, and/or imaging, the radar is often designed to maximize SNR. Maximizing SNR can be accomplished by applying a filter to the received signal that retains the desired signal while suppressing all unwanted noise and interference. A matched filter is one that is matched to the range and velocity of a target to maximize the SNR and improve the probability of detection [31]. The filter is considered matched because it is optimally tailored to the specific receive signal it is filtering. Unfortunately, the range and velocity of the target are seldom known *a priori*, so a bank of filters matched to all possible ranges and velocities must be used to find the right match. The output of the filter, matched to the narrowband return signal in both range and Doppler, is defined as [13]

$$y(\tau, f_d) = \int_{t=0}^{T} s_R(t)s^*(t-\tau)e^{-j2\pi f_d t}dt, \tag{2.15}$$

and is often referred to as the range-Doppler correlation function. The wideband range-Doppler correlation function is given by

$$y(\tau_{ref}, \alpha_{ref}) = \int_{t=0}^{T} s_R(t)s^*(\alpha_{ref}(t-\tau_{ref}))dt, \tag{2.16}$$

where τ_{ref} is the reference delay used to estimate the range between the radar and the target and α_{ref} is the reference time-scale used to estimate the relative motion between the radar and the target.

2.4.3 Ambiguity Function.

Ambiguity functions are the mathematical tool used by radar designers to evaluate a waveform's range and velocity resolutions simultaneously. These functions characterize the response of the matched filter by describing the behavior of the radar correlation over time for all range and velocity values [7]. In other words, the ambiguity function is used to examine the ambiguities, or uncertainties, of the system in range and velocity. The ambiguity function is defined as the magnitude of the range-Doppler correlation function [16] of (2.15), and is expressed in its narrowband form as

$$\chi(\tau, f_d) = |y(\tau, f_d)| = \left| \int_{t=0}^{T} s_R(t) s^*(t-\tau) e^{j2\pi f_d t} dt \right|, \tag{2.17}$$

where $s_R(t)$ was previously defined in (2.12). The generalized wideband ambiguity function is similarly defined as [5]

$$\chi(\tau_{ref}, \alpha_{ref}) = \left| \int_{t=0}^{T} s_R(t) s^*(\alpha_{ref}(t-\tau_{ref})) dt \right|. \tag{2.18}$$

However, following Axelsson's notation in [5], a change of variables is required. Setting $u = \alpha_{ref} t - \tau_{ref}$ and thus $t = (u + \tau_{ref})/\alpha_{ref}$, the integral in (2.18) becomes

$$\int s(\alpha t - \tau) s^*(\alpha_{ref}(t - \tau_{ref})) dt = \int s\left(\frac{\alpha}{\alpha_{ref}} (u + \tau_{ref}) - \tau \right) s^*(u) dt. \tag{2.19}$$

Setting $\beta = \alpha/\alpha_{ref}$ and $\Delta\tau = \tau - \beta\tau_{ref}$, the generalized wideband ambiguity function can be represented as

$$\chi(\beta, \Delta\tau) = \left| \int_{t=0}^{T} s(\beta t - \Delta\tau) s^*(t) dt \right|. \tag{2.20}$$

The formula in (2.17) is often referred to as the Woodward ambiguity function because of Woodward's research in comparing range and Doppler performance of a number of waveforms [41]. Often the ambiguity function, $\chi(\tau, f_d)$, is represented graphically to view the performance and associated trade-offs of the system in both range and Doppler [27]. Figure 2.6 shows the ideal thumbtack ambiguity response that results from the random, aperiodic noise waveform.

The random noise radar ambiguity function has been studied in great detail. In 1966, Rihaczek [32] extended the wideband ambiguity function to bandwidths and ranges that were not presented by Woodward. Rihaczek found that the relative radial velocity, v, of a target did not simply result in a time dilation of the signal but also in an amplitude change by a factor of $1 + v/f_0$, where f_0 is the mean frequency of the wideband signal. This amplitude change is small at practical velocities and is often ignored. For the purposes of this research effort, the velocities are sufficiently small and the resulting amplitude change will be ignored.

Axelsson [5], along with Dawood and Narayanan [7, 8] have provided much of the recent research in UWB random noise radar ambiguity functions. Their work in generalizing the ambiguity functions for UWB random noise waveforms has provided the building blocks for determining the AFIT RNR ambiguity function that will be discussed in Section 2.5.1.

2.4.4 Resolution.

The ambiguity function of (2.17) can be used to examine the limiting relationships between the measurement window, T, velocity, v, and range resolution, ΔR, defined in (2.8) [4]. A velocity ambiguity results if the time a moving target takes to pass through a range resolution cell is more than the integration time. This means $\Delta R/v = c/(2Bv)$ must be greater than T to avoid velocity ambiguity. The signal length, N, is shown in (2.13) to equal $2BT$ resulting in an upper limit on N defined as [4]

$$N = 2BT < \frac{c}{v}. \tag{2.21}$$

From (2.21), it can be seen that the signal length must be shorter than the ratio of the speed of light to the velocity of the target. As target velocity increases, the signal length upper bound is shortened.

2.5 AFIT Random Noise Radar

The AFIT RNR, also known as the AFIT Noise Network (NoNET) when more than one RNR is networked, is designed to produce highly resolved imagery of a target scene while maintaining the LPI characteristics inherent in UWB noise radars [27]. It can be configured in a monostatic or multistatic mode as well as a netted monostatic mode that simply shares target information between multiple monostatic nodes.

In order to keep the design simple and configurable, the AFIT RNR consists of a radio frequency (RF) front end, and a digital receiver known as the direct conversion receiver (DCR). The ADC samples the RF transmit and receive signals without mixing to baseband, thus providing a direct conversion of the RF signals for correlation processing. A block diagram of the AFIT RNR can be seen in Figure 2.8. The RF front end consists of a thermal noise generator, a transmit and receive antenna, two band pass filters (BPF) and two low noise amplifiers (LNAs). Its thermal noise generator produces a transmit waveform subsequently limited to a bandwidth of approximately 400 MHz with a high frequency of 800 MHz. The voltage signal has a Gaussian amplitude distribution where the power spectral density is nearly uniform across the bandwidth at approximately -85 dBm/Hz. Both the transmit and receive channels use log-periodic antennas (LPAs) that are not ideal and introduce a nonuniform frequency response. The receive signal is filtered and amplified using the LNAs before it is sent to the ADC.

The DCR begins with two-channel ADC that samples the transmit and receive channels at 1.5 Gsamp/s each. The digital signals are then correlated using a number of MATLAB® subroutines that have been developed for each of the AFIT RNR configurations.

In addition to the hardware system, a model was developed in Simulink to simulate the entire radar from end to end [30]. The model emulates the radar system, allowing

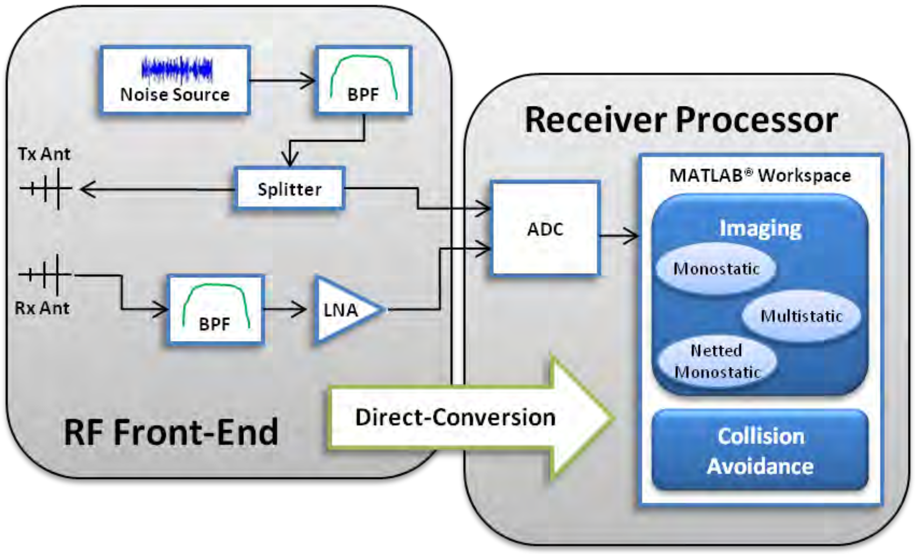


Figure 2.8. The AFIT NoNET consists of a RF front end and a direct conversion receiver.

flexibility to digitally modify components in the radar and examine the effects the modifications have on radar performance.

2.5.1 Simultaneous Range and Velocity Processing.

The most common approach to velocity estimation in noise radar literature is based on Narayanan's heterodyne receiver and classical Doppler processing [25]. There are two basic components required to implement this classical Doppler processing. First, the phase information of the signal must be known, and second, the a narrow-band noise model must be assumed.

Because Doppler radars estimate target velocity by comparing the frequencies of the transmit and receive signal, phase coherence must be maintained. To introduce this phase coherency, a local oscillator (LO) is used for mixing and frequency conversion of both the transmit and receive signal. The resulting in-phase (I) and quadrature (Q) channels provide the phase coherence required to measure the Doppler shift of the receive signal.

Additionally, classical Doppler processing requires a narrowband assumption. Translating Doppler frequency shift, f_d, to relative radial target velocity, v, is done using (2.2). In an UWB radar, the transmit signal is not defined by a single frequency (with a single wavelength λ), but rather by a range of frequencies from f_l to f_h as defined by the 3 dB bandwidth. So, unless all frequencies within the signal experience the same Doppler shift, there will be an error when using (2.2) to estimate target velocity. One method to mitigate such errors is to average the transmit waveform with center frequency f_0 over long intervals which results in the Doppler equation [25]

$$f_{d0} = \frac{2v}{\lambda_0} \cos\psi, \tag{2.22}$$

where f_{d0} and λ_0 correspond to the mean Doppler frequency and mean transmit wavelength, respectively.

Because the AFIT RNR uses a digital receiver in place of Narayanan's heterodyne receiver design, and because the AFIT RNR transmits over a large fractional bandwidth with no carrier frequency, classical Doppler analysis results in untenable errors in the AFIT RNR [2]. Lievsay and Akers [19], however, proposed a method to extract velocity information using the AFIT RNR's phase-incoherent digital correlation. The transmitted noise waveform is sampled at $f_s = 2f_h$, where f_h is the waveform's maximum frequency. Target velocity causes each sample of the receive signal to shift in time by Δt relative to the received signal from a stationary target. Figure 2.9 illustrates how the relative radial velocity of a target, v, causes the received measurement window, T_{rx} to differ from the transmit measurement window, T_{tx} by

$$T_{rx} = T_{tx}/\alpha, \tag{2.23}$$

where α is the time scale of the receive signal as defined in (2.11).

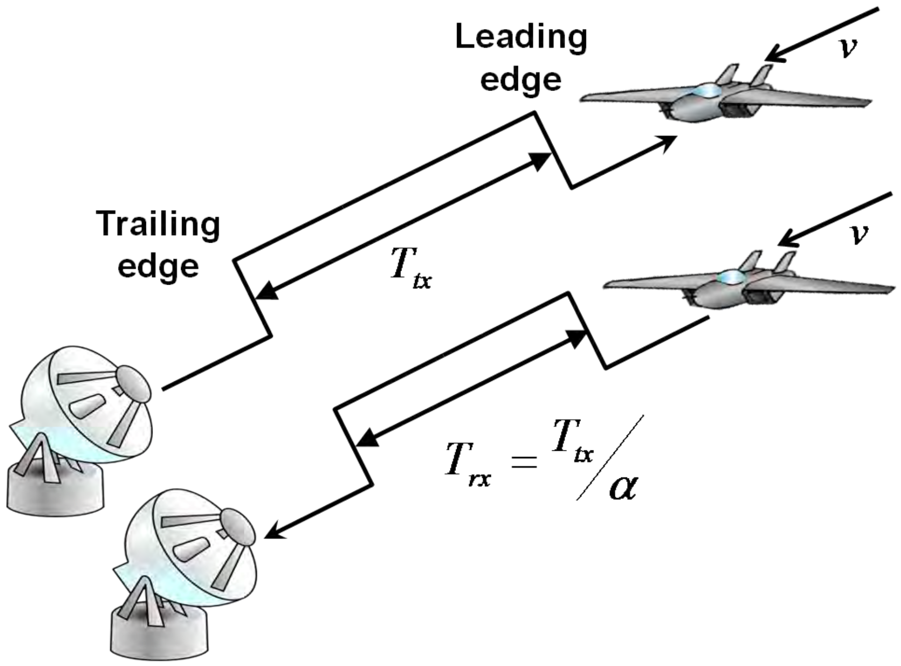


Figure 2.9. Illustration of the time domain velocity estimation technique implemented in the AFIT RNR.

The overall difference between the duration of the receive signal and the duration of the transmit signal due to the target velocity, v, is given by $\Delta T = T_{rx} - T_{tx}$ as can be seen in Figure 2.10. The ΔT of the overall measurement window results from the relative time shift at each sample, expressed as

$$\Delta t = \frac{2v}{(c-v)f_s}. \tag{2.24}$$

From this relationship, a target's radial velocity can be derived by measuring the time shift at each sample over the length of the measurement window. It is important to note that this method assumes the target's radial velocity is constant over the measurement window T_{tx}.

Lievsay [18] created a bank of reference signals, analogous to Doppler filter banks,

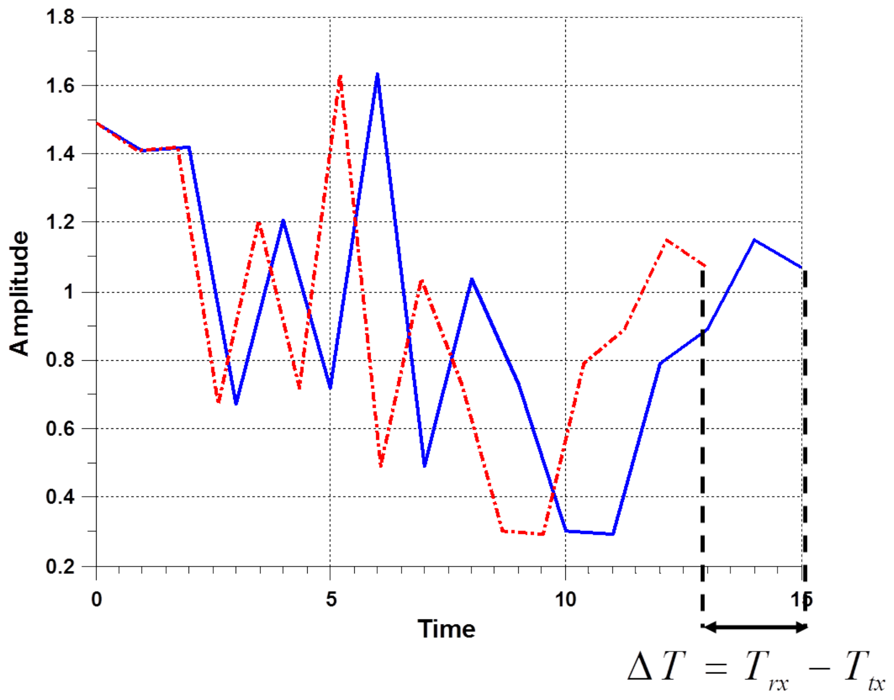


Figure 2.10. Ingressing radial velocity shortens the measurement window by ΔT.

in the form of

$$s_{ref}[k] = s[k - (k-1)\Delta t], \tag{2.25}$$

where each reference signal in the bank has a Δt that corresponded to a set reference velocity. Each reference signal is then cross-correlated with the measured signal, and based on a single-target assumption, the signal with the highest correlation corresponded to the estimated velocity of the target. The reference signal bank must be chosen carefully. If the velocity increments are too large, the radar may miss the target due to the thumbtack ambiguity response. However, each velocity increment requires processing time and memory. Increments that are too small would require more reference signals and could quickly overwhelm memory, resulting in prohibitively long processing times. Unfortunately, the bank of reference signals cannot be generated *a* *priori* because the transmitted signal is random for each measurement window.

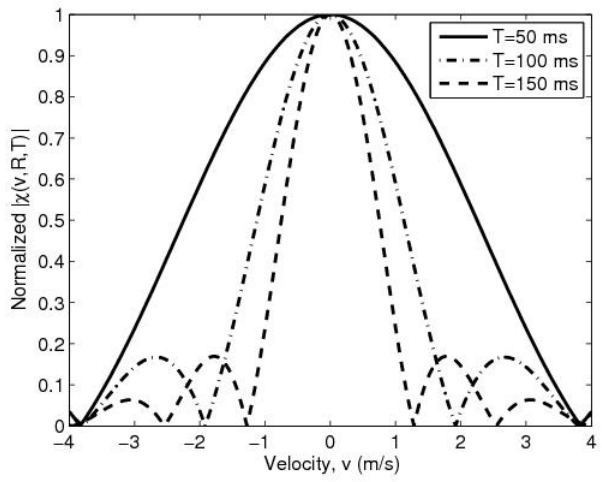
The length of a transmit and receive signal is given by $N = \lceil f_s \cdot T \rceil$, where $\lceil \cdot \rceil$ represents the integer ceiling of the computed value. Velocity resolution is directly tied to the highest frequency of the signal, f_h and the measurement window, T. This relationship can be seen by analyzing the wideband ambiguity function, defined by Axelsson [5], and applied to the AFIT RNR by Lievsay [18]. The AFIT RNR wideband ambiguity function is an expansion of (2.18) given by

$$|\langle \chi(\Delta\tau, \beta, t) \rangle| \;=\; \left| \int_t^{t+T} \operatorname{sinc}\left[\pi B((\beta-1)\zeta - \Delta\tau)\right] \cos(j2\pi f_c(\beta-1)\zeta) d\zeta \right|, \quad (2.26)$$

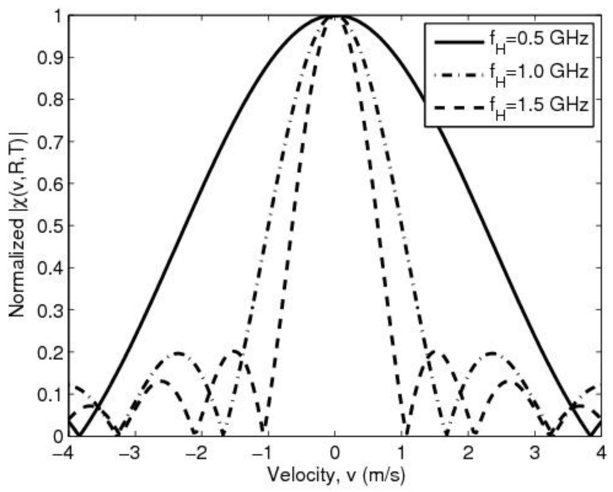
where B is the signal bandwidth limited at the upper end by f_h, β is the relative time scale given by $\beta = \alpha/\alpha_r$, and $\Delta\tau = \tau - \beta\tau_r$. From (2.26), it can be seen that the measurement time, T, and the bandwidth, B, define the velocity resolution within the ambiguity function. The ambiguity function can also be represented graphically to view velocity and range performance tradeoffs. An example of an ambiguity function plot can be seen in Figure 2.6.

As illustrated in Figure 2.11, velocity resolution is improved as f_h or T increases. However, the computational requirements of the DCR increase as well. For a sampling frequency of 1.25 GHz and a measurement window of 160 ms, the signal to be stored in a processor's random access memory (RAM) has 200 million samples, making the amount of available memory a limiting factor. Due to processing limitations, Lievsay limited the signal length to 200 million samples. An increase in measurement time would result in a decrease to the sampling frequency to maintain $N = \lceil f_s \cdot T \rceil = 200$ million. At a sample rate of 1.25 Gsamp/s, the measurement window that resulted in a signal length of 200 million was set to 160 ms.

In addition to limiting the length of the signal for his experiment, Lievsay [18] limited the span of velocities to be measured and the velocity resolution. He measured



(a)



(b)

Figure 2.11. (a) As the measurement window, T, grows, the velocity resolution improves. Similarly, (b) as the highest signal frequency grows, the velocity resolution improves. (From [18])

velocities spanning only -2 to -14 m/s, with increments of 0.5 m/s. This velocity span corresponded to 25 reference signals in the form of (2.25), each with a different Δt. Each reference signal was correlated with the measured receive signal and plotted in a 2D range-velocity plot. With these limitations, the simultaneous range-velocity processing using actual measured data for an inbound target at 10 meters traveling at 5 m/s was 42 minutes, which is significantly greater than the measurement window of 160 ms.

The RAM required using Lievsay's [18] velocity estimation technique was 32 GB, making parallel processing using AFIT's equipment impossible. However, if the memory requirement is sufficiently reduced, the processing can be done in parallel using multiple CPUs or even using an FPGA or GPU. To reduce the memory requirement, a single-bit (or binary) ADC can be used in place of the existing eight-bit ADC. Axelsson [3] has shown that, for a single target with a high SNR, a binary ADC will perform as well as a high resolution ADC for range and velocity processing.

2.5.2 AFIT RNR Signal-to-Noise Ratio.

The noise radar RRE varies from the single pulse (or single sample) RRE of the pulse radar given in (2.5). The maximum range is affected by the integration gain of the correlation receiver. The integration gain, BT, is also known as the time-bandwidth product and, when applied to the RRE, leads to a the noise radar RRE given by

$$R_{max} = \sqrt[4]{\frac{P_t G_t G_r \lambda^2 \sigma T}{(4\pi)^3 k T_0 F L_s (\text{min SNR})}}. \tag{2.27}$$

Furthermore, including the integration gain into the noise radar SNR leads to the equation

$$SNR_{RNR} = \frac{P_r}{P_n} = \frac{P_t G_t G_r \lambda^2 \sigma T}{(4\pi)^3 R^4 k T_0 F L_s}. \tag{2.28}$$

Integration time results in an additional design factor for noise radar engineers. Not only will increasing power and antenna gain improve the maximum detectable range, but range will also be improved by increasing integration time.

2.5.3 Pre-Processing SNR.

As seen in (2.28), P_t, G_t, G_r, λ, σ, T, R, B, F, and L_s are the only variable parameters affecting the SNR of the radar. Typically, the target RCS and range are out of the hands of the radar engineer and the loss figure is specific to the application and environment, narrowing the list of modifiable parameters. In the case of the AFIT RNR, the high and low frequencies are set, holding λ and B constant as well. Only P_t, G_t, G_r, T, and F are left available for the radar engineer to impact SNR. Of these parameters, F is dependent on the components in the receive path, and until those components are miniaturized in the future, the noise figure is assumed constant over the signal bandwidth. Similarly, because of the flat power spectral density of the noise source as discussed in Section 3.2.1, the transmit power, P_t, can be assumed to be constant over the signal bandwidth as well.

Without significant hardware changes to the AFIT RNR, the transmit and receive antenna gain and the integration time are the only parameters that can impact the SNR of the system. The current antennas do not provide a constant gain across the signal bandwidth. The frequency dependence of the antenna gain can have a significant impact on the SNR. Ludwig [20] is designing a new antenna for the AFIT RNR that will have a smaller form factor and likely have a more constant gain across the signal bandwidth.

Integration time, as seen in (2.28), is directly proportional to the SNR of the AFIT RNR. Although the parameter is modifiable, the measurement window is currently limited to 1 μs due to the hardware constraints of the ADC. Extending the measurement window past 1 μs will require a new digitizer board. Although modifiable to improve SNR, there is no frequency dependence of the integration time on the overall SNR of the system.

Frequency dependence is further introduced into the AFIT RNR by the LNAs.

The LNA frequency response is illustrated in Figure 2.12. Approximately 2 dB of gain separates the lower end of the band of interest from the upper end. Two LNAs are cascaded in the AFIT RNR, leading to a change of roughly 4 dB within the passband. It is expected that the system noise is white Gaussian, thus not a function of frequency. The noise, however, is amplified in the LNAs along with the receive signal resulting in a frequency dependent noise signal.

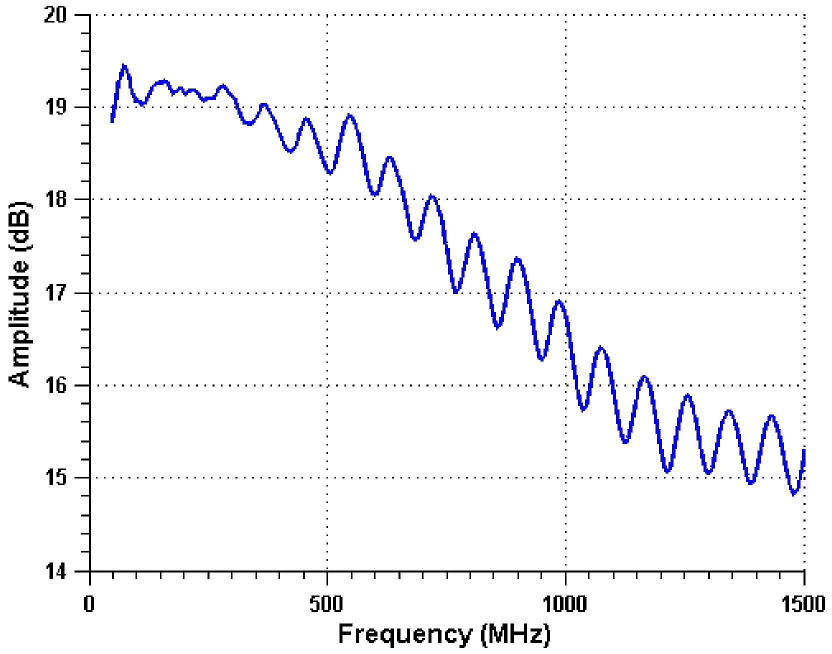


Figure 2.12. Each LNA in the AFIT RNR results in a 2 dB disparity across the passband (400-800 MHz).

2.5.4 Post-Processing SNR.

The DCR that has been implemented in the AFIT RNR has a significant impact on the SNR of the system. To begin, the ADC introduces an error between the analog input and the digital output known as quantization noise. Second, the correlation process results in a peak response as well as a correlation floor also known as the sidelobe level.

The eight-bit ADC used in the AFIT RNR has been studied in great detail by Nelms [27], who concluded that the quantization noise spectra of the ADC is nearly uniform, causing no significant cross-correlations that can lead to erroneous target estimations. The quantization noise of the ADC does, however, add to the noise figure of the receiver, which reduces the SNR and can negatively impact the probability of false alarm.

In [39], Walden states that the SNR (in dB) of an ideal ADC can be defined as

$$SNR = 6.02N_b + 1.76, \tag{2.29}$$

where N_b is the number of bits in the ADC. For a given input signal, the noise power resulting from an eight-bit ADC is approximately 50 dB less than the signal power. Conversely, for the binary ADC discussed in Section 3.3.1, the noise power resulting from the ADC is only 7.78 dB less than the signal power.

Axelsson discusses the correlation processing in the range dimension of a wideband noise signal in [2, 3]. Following his notation, range correlation in its simplest form, is a comparison of the noise signal $s[k]$ with a time-delayed version $s[k-m]$ and represented as

$$r[m] = \sum_{k=0}^{N-1} s[k]s[k-m], \tag{2.30}$$

where N is the length of the signal defined by the integration time. The peak of the correlation function occurs when perfectly matched in time ($m = 0$) and its amplitude is defined as

$$r[0] = N < s^2[k] >= N\sigma_r^2, \tag{2.31}$$

where σ_r^2 is the variance of the received signal and $< \cdot >$ represents the signal mean. The squared correlation floor, also known as the sidelobe variance σ_s^2, is found when $m \neq 0$ and given by

$$\sigma_s^2 = \sum_{k=0}^{N-1} < s^2[k] >< s^2[k-m] >= N\sigma_r^4. \tag{2.32}$$

Combining (2.31) and (2.32), the ratio between the squared correlation peak resulting from a matched range and the sidelobe variance resulting from mismatched ranges can then be defined as

$$\frac{r^2[0]}{\sigma_s^2} = \frac{N^2\sigma_r^4}{N\sigma_r^4} = N. \tag{2.33}$$

This equation is defined as the peak-to-average sidelobe ratio and can also be considered the post-processing SNR.

The pre-processing SNR plays a role in the correlation peak-to-average sidelobe ratio. To see the role of the pre-processing SNR (σ_r^2/σ_n^2), additive noise must be included in the derivation above. The correlation function with noise can be represented as

$$r[m] = \sum_{k=0}^{N-1} (s[k]s[k-m] + s[k]w[k]), \tag{2.34}$$

where $w[k]$ represents the additive Gaussian noise with variance, σ_n^2, used to determine the SNR of the pre-processed signal. The presence of noise does not change the correlation peak amplitude but does increase the sidelobe variance to

$$\sigma_s^2 = N\sigma_r^4 + N\sigma_r^2\sigma_n^2. \tag{2.35}$$

Continuing, the peak-to-average sidelobe ratio with additive noise becomes

$$\frac{r^2[0]}{\sigma_s^2} = \frac{N^2\sigma_r^4}{N\sigma_r^4 + N\sigma_r^2\sigma_n^2} = \frac{N}{1 + \mathrm{SNR}^{-1}}. \tag{2.36}$$

However, there are a number of factors that complicate this analysis. First the signals that are correlated are not exact replicas. The target environment, noise, hardware components, and other interference cause changes to the receive signal. These changes effect the pre-processing SNR of the signal and are part of the cross-correlation processing.

2.6 Chapter Conclusion

The AFIT RNR is a low-power, electromagnetically compatible, and flexible system originally designed for high resolution radar imagery. It also has a number of characteristics that make it ideal as an autonomous collision avoidance sensor. This chapter presented the basic principles of noise radar technology. The next chapter will tie these principles into the research effort to demonstrate the AFIT RNR's simultaneous range and velocity processing and the software model used for the SNR analysis.

III. System Description and Methodology

3.1 Chapter Overview

To investigate the AFIT RNR's ability to simultaneously process range and velocity information in minimal time, a logical and procedural effort was established. Those procedures are presented in this chapter along with an overview of the hardware and software used in the simultaneous processing. Additionally, the methodologies used to build the AFIT RNR software model are presented, allowing for a comparison of the theoretical capabilities of the RNR with measured results, and providing the foundation for a thorough analysis of the noise radar's SNR.

3.2 System/Equipment Description

The AFIT RNR was first constructed by Schmitt [33] and was demonstrated in near-monostatic and networked configurations. The bistatic/near-monostatic configuration, broken into its functional blocks, can be seen in Figure 3.1, and is the focus of this research effort.

3.2.1 Transmitter.

To generate a random noise transmit signal, the AFIT RNR uses a thermal noise generator developed by Noise Comm®. This white Gaussian noise source provides a flat response at -82 dBm/Hz up to 1.6 GHz. The source is then filtered using a low-pass filter (LPF) and a high-pass filter (HPF) to generate the band-limited signal from 400 to 800 MHz. After filtering, the noise signal is split to the transmit antenna as well as to the direct conversion receiver (DCR), where it is used as a reference signal for correlation processing. The transmit and receive antennas are wideband

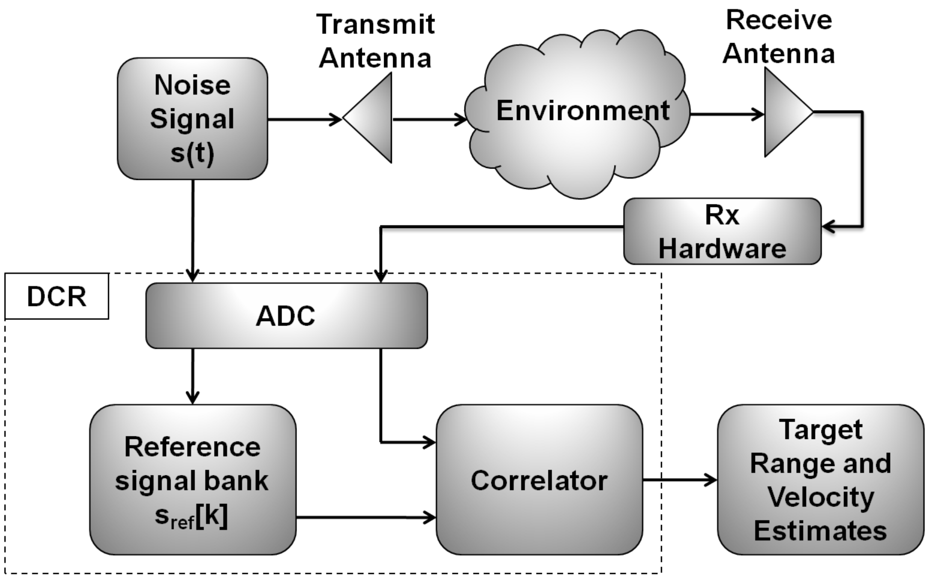


Figure 3.1. This figure illustrates the functional building blocks of the AFIT RNR in its bistatic/near monostatic configuration.

log-periodic antennas (LPAs) that offer frequency dependent gain in the ballpark of 6 dB. The transmit path of the AFIT RNR can be seen in Figure 3.2.

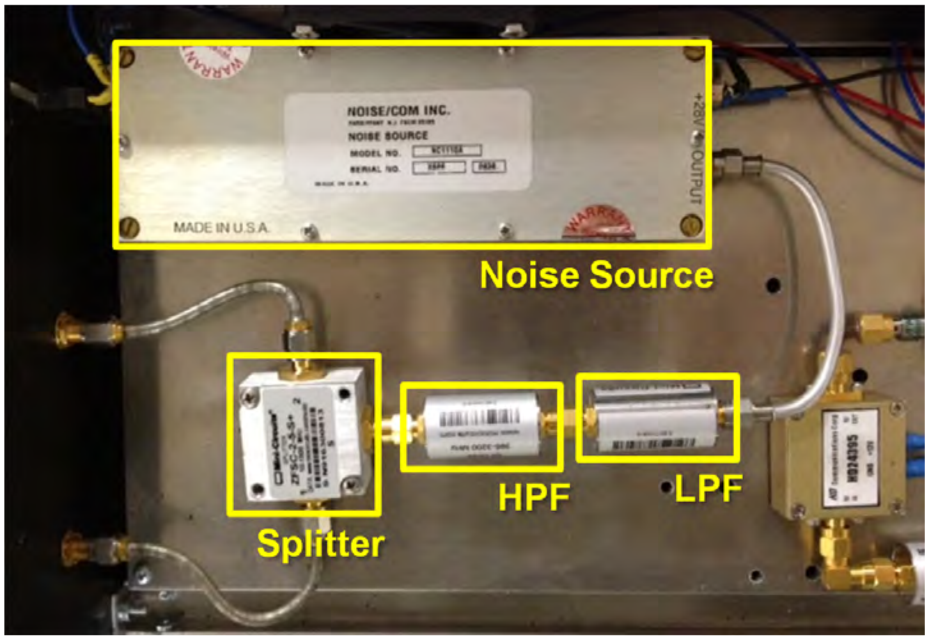


Figure 3.2. This figure highlights each component in the AFIT RNR transmit path.

3.2.2 Receiver Front End.

After the transmit signal interacts with the environment, the return echo at the receive antenna experiences a similar gain as experienced by the transmit signal at the transmit antenna. The signal is then passed through a LPF and HPF combination, identical to the transmit path filters, before being amplified by two, 20-dB low noise amplifiers (LNAs). The LNAs are used to bring the receive signal amplitude within the dynamic range of the ADC. The receive path of the AFIT RNR can be seen in Figure 3.3, and a detailed description of each hardware component in the AFIT RNR can be found in [33].

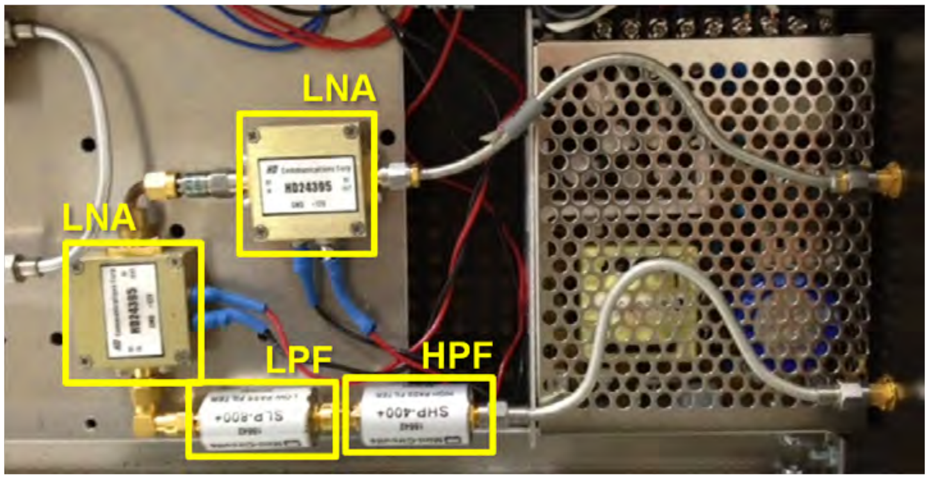


Figure 3.3. This figure highlights each component in the AFIT RNR receiver front end (prior to ADC).

3.2.3 Direct Conversion Receiver.

The DCR performs the transmit and receive signal analog-to-digital conversion as well as the correlation processing required for range and velocity estimation using the theory described in Sections 2.4 and 2.5. The ADC is developed by Acquisition Logic® and has a bit-depth of eight bits. The single-channel sampling rate of the ADC is 3 Gsamp/s, while the two-channel sampling rate is 1.5 Gsamp/s with a maximum acquisition time of 1 μs. Although limited to the two-channel 1.5 Gsamp/s sampling rate, the ADC can interpolate up to three additional samples per measured sample to bring the effective two-channel sampling rate to 6 Gsamp/s in the standard AFIT RNR configuration.

The ADC is connected to a Dell Inspiron 640m laptop through a Peripheral Component Interconnect Express (PCIe) card and uses direct memory access to place the digital signals directly into the MATLAB® workspace. With the transmit (or reference) and receive signals in digital format, the correlation processing takes place using a MATLAB® routine developed by Schmitt [33] but further refined by Nelms and Priestly [27] and [30]. This algorithm uses a 1 μs measurement window and performs only range correlation. The AFIT RNR's ability to measure range precisely has been documented in [27] and [30]. Velocity estimation in the standard AFIT RNR application has been based on this range estimation capability and exploits the well known equation

$$velocity(v) = \frac{distance(d)}{time(t)}. \tag{3.1}$$

Given two measurement windows separated by time t, and assuming a constant target velocity, the relative radial velocity of the target, v, can easily be calculated. This velocity estimation technique, however, requires multiple measurements to determine target velocity. To perform simultaneous range and velocity processing within the DCR a new algorithm was developed by Lievsay [18]. As discussed in Section 2.5.1, the two-dimensional (2D) processing approach worked, but the processing times and hardware required were prohibitive for any practical application.

3.3 2D Processing Performance

With the initial work accomplished by Lievsay [18], a need to develop an efficient 2D processing algorithm was uncovered. The 42-minute processing time and 32 GB RAM requirement made the 2D processing impractical for AFIT RNR applications. This section describes the two-fold approach used to bring the 2D processing application nearer to practical implementation. The first approach attempts to reduce the memory required for 2D processing by simulating a binary ADC in place of the eight-bit ADC. The second approach involves segmenting the Fast Fourier Transforms (FFTs) used for correlation in order to parallelize the correlation processing.

3.3.1 Binary ADC.

As discussed in [18], Lievsay implemented the 2D processing algorithm in a controlled scene with a single inbound target at a range of 10 m and a velocity of -5 m/s. The RAM required using this velocity estimation technique was 32 GB, making parallel processing using AFIT's equipment impossible. However, if the memory requirement is sufficiently reduced, the processing can be done in parallel using multiple central processing units (CPUs) or even using a field-programmable gate array (FPGA) or graphics processing unit (GPU). To reduce the memory requirement, a single-bit (or binary) ADC can be used in place of the existing eight-bit ADC.

Axelsson [3] has shown that, for a single target with a high pre-processing SNR, a binary ADC will perform as well as a high resolution ADC for range and velocity processing. Further, he has shown that multiple targets can lead to reduced sidelobe suppression in a binary ADC compared to an ideal ADC. To improve sidelobe suppression, a secondary noise signal can be added to the received signal before the ADC. The added noise signal can reduce the sidelobe amplitudes to those of an eight-bit ADC at the expense of the pre-processing SNR. For the purposes of this exercise, only a single target is present, and the post-processing SNR is expected to remain the same for the 2D correlation using only the signs of the received signal vice the full eight-bit ADC signal values.

In an effort to compare the results of the binary signal to the results measured by Lievsay [18], Lievsay's measured data for a single inbound target with known velocity was used for the MATLAB® simulation outlined here. Because of the long measurement windows required for adequate velocity resolution ($\approx$ 150 ms), the ADC used in the DCR, which has an acquisition limit of 1μs, could not be used. Instead, the original data was collected by Lievsay using the AFIT RNR with a Tektronix® Digital Phosphor Oscilloscope (DPO) 7254 as the eight-bit ADC. The dataset that is used for the test is of a target at 10 m from the monostatic radar moving directly toward the RNR at 5 m/s. The measurement window was 160 ms and sampled at $f_s = 1.25$ Gsamp/s.

The test is conducted in an eight-step process using a single computer with the specifications given in Table 3.1. In each step, the processing time and the peak instantaneous memory usage are recorded, and the results are compared to those attained using Lievsay's method. The test procedure is:

1. Replicate Lievsay's results using his data and correlation algorithm.
2. Optimize Lievsay's correlation algorithm without changing its functionality.
3. Convert transmit and receive signal from double to single precision.
4. Convert transmit and receive signal from single precision to sign only (+1 or -1) to simulate the output of a binary ADC.
5. Replace *interp1* function to reduce processing time and memory.
6. Use multiple (four or more) processors on single computer.

7. Change the number of points in the FFT used for signal correlation to a power of two in order to speed up the *fft* function.

8. Evaluate the possibility of using a GPU to speed up processing time.

3.3.2 FFT Segmentation.

As discussed in Section 3.2 and illustrated in Figure 3.1, the transmit signal, $s(t)$, is split and passed into the DCR, where a bank of reference signals is generated based on the transmit signal and set of pre-defined reference velocities. Based on (2.24), the selected reference velocity shifts each of the N samples of the transmit signal by Δt to give a reference signal in the form of (2.25).

The receive signal is then correlated with each of the reference signals. The cross correlation function, $r(\tau)$, is defined as [13]

$$r(\tau) = \int_{t=0}^{T} s_R(t) s_{ref}(t - \tau) dt, \tag{3.2}$$

where $s_R(t)$ is the receive signal and $\tau = 2R/c$ represents the range to the target in terms of the time delay. However, in the DCR, the correlation is performed digitally in the form of

$$r[m] = \sum_{k=0}^{N-1} s_R[k] s_{ref}[k - m], \tag{3.3}$$

where $m = f_s \tau$ corresponds to the number of samples for delay τ. To implement this cross correlation function in an efficient manner, the FFT is used.

Correlation is similar to convolution, and that similarity can be exploited to take advantage of the FFT efficiencies. Convolution in the time domain is equivalent to multiplication in the frequency domain. The difference between convolution and correlation is simply that in convolution, the reference signal is reversed in the time

Table 3.1. Processing Computer Specifications

Make	Hewlett-Packard
Model	Z800 Workstation
Operation System	Windows 7 Pro
Processor Make	Intel
Processor Model	Xeon X5667
Number of Processors	8
Processor Speed	3.07 GHz
64-bit Technology	Yes
Installed Memory	48 GB

domain, which is not the case in correlation. That reversal is equivalent to a conjugation in the frequency domain, leading to the equation for correlation

$$r[m] = \frac{1}{N}\mathcal{F}^{-1}\left[\mathcal{F}\{s_R[k]\}\mathcal{F}\{s_{ref}[k]\}^*\right], \tag{3.4}$$

where $\mathcal{F}$ represents the Fourier transform, $\mathcal{F}^{-1}$ represents the inverse Fourier transform, and * represents conjugation.

Although the FFT is efficient, the lengthy signals required for sufficient velocity resolution, and hence the lengthy FFTs, are too long to allow for parallel implementation in a GPU. The signals must be broken into small segments, thus reducing the FFT sizes to allow for parallelization over hundreds of processors.

Meller published a method in [24] to segment lengthy FFTs in a noise radar correlator. More commonly known as the overlap-save method [28], the FFTs are broken into overlapping segments of length $2M$, where M is equivalent to the number of samples in the time delay corresponding to the range extent R_{max}. The receive signal segments have M samples from $s_R[k]$ and are padded with M zeros to have a segment length of $2M$ samples. The reference signal segments, on the other hand, are a concatenation of M samples from the "previous" segment and M samples from the "current" segment, thus overlapping the FFT segments.

Once the signals have been segmented, the FFTs of the receive signal and reference signal segments are computed. The conjugated reference signal FFT segment is then multiplied with the receive signal FFT segment, and the inverse FFT (IFFT) of the result is computed. The segmented IFFTs are then accumulated (the vectors are added), resulting in the cross correlation of the reference and receive signal. A comparison of the the traditional cross correlation implementation with the segmented method proposed by Meller [24] can be seen in Figure 3.4.

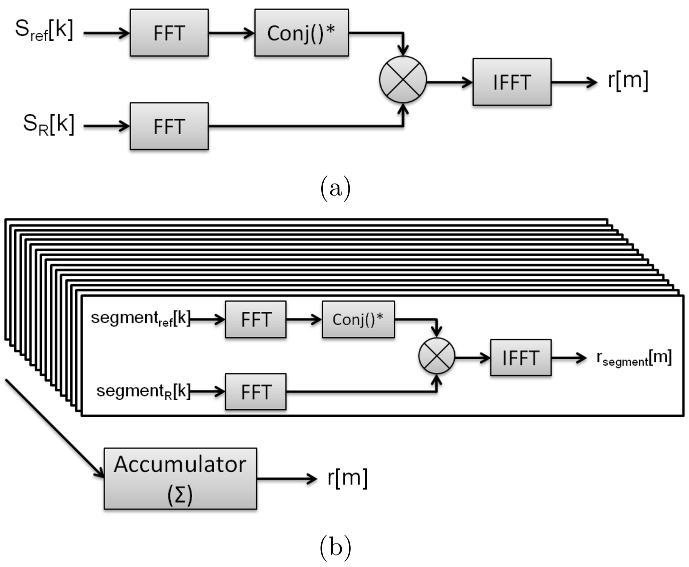


Figure 3.4. Comparison of (a) the traditional cross correlation implementation and (b) the segmented cross correlation method proposed by Meller [24].

The benefit of this FFT segmentation method is its potential for parallelization. The cross-correlation of each segment can be computed individually and in parallel before accumulation. Instead of a single cross correlation that has FFTs of length 200 million, there can be many (thousands) of cross correlation operations that take the place of the single operation. These fast operations can be implemented on a GPU or FPGA and distributed to hundreds of processing cores operating in parallel, thus significantly reducing the overall processing time.

Two computers, equipped with NVIDIA® GPUs, were used to process the collected data. The specifications for each of the computers along with the GPUs are presented in Table 3.2.

The test procedure for this element of the test effort is as follows:

1. Update the algorithm to include FFT segmentation. Determine processing time on both multi-core PCs without GPU computing.

2. Modify the algorithm for GPU computing using MATLAB®'s GPU interface. Determine processing time on both GPU equipped multi-core PCs.

3. Modify the algorithm for GPU computing using Jacket®'s GPU interface. Determine processing time on both GPU equipped multi-core PCs.

MATLAB® has developed a GPU interface as part of the parallel computing toolbox. A number of GPU specific commands have been created to pass CPU variables to the GPU to perform computations on the GPU and then gather the results back to the CPU.

Another company, AccelerEyes®, has developed a product called Jacket® that claims to be better than the parallel computing toolbox in MATLAB®. Jacket® allows MATLAB® users to interface with the GPU without getting into the low-level programming details. Jacket® supports many MATLAB® functions to make modifying existing algorithms for GPU computing fairly seamless. Both Jacket® and MATLAB®'s parallel computing toolbox will be used to find the best solution to simultaneous range and velocity processing in the AFIT RNR.

3.4 Software Model

To build credibility into the SNR analysis, a robust model of the AFIT RNR is required to simulate the expected behavior of the system. Priestly created a basic model in Simulink® [30] that provides the foundation for the robust model required

Table 3.2. **Parallel Processing Hardware**

	Computer 1	Computer 2
Make	Dell	HP
Model	Precision T7500	Z8000 Workstation
Operating System	Windows 7 Pro	Windows 7 Pro
Processor Make	Intel	Intel
Processor Model	Xeon W5590	Xeon X5667
Number of Processing Cores	4	8
Processor Speed	3.33 GHz	3.07 GHz
Installed Memory	48 GB	48 GB
GPU Make	NVIDIA	NVIDIA
GPU Model	Tesla 1060	Tesla C2070
GPU Processing Cores	240	448
GPU Shared Memory	4 GB	6 GB

for SNR analysis. This model along with the significant updates required are discussed in this section.

3.4.1 Previous Work.

Each hardware component in the AFIT RNR can be modeled in the Simulink® environment using the specifications given in the component data sheets and summarized by Schmitt in [33]. Some components are easier than others to model. For instance, the LPFs and HPFs can be modeled fairly easily and accurately by modifying the parameters in the respective Simulink® blocks to match the component data sheets. Similarly, the LNAs and the noise source can be modeled accurately without too much trouble. The log-periodic antennas and the target environment, however, are more difficult to model. Priestly [30] laid the framework for the entire model, but did not have time in his research effort to properly characterize each element in the model, specifically in the antenna and target environment subsystems.

3.4.2 Necessary Changes.

The first significant change to the model is the transmit and receive antenna subsystems. The LPAs do not provide a consistent gain across the entire bandwidth of the transmit signal; thus a standard 6 dB gain is not adequate. Ludwig [20] measured the reflection curve (S11) of a representative LPA and the results can be seen in Figure 3.5. Although this curve is not a measure of antenna gain, it is clear that the losses due to reflection vary widely with frequency. Assuming the other losses in the antenna are minimal and whatever is not reflected in the antenna is transmitted, a better model of antenna gain would be frequency dependent and proportional to the inverse of the reflection curve, as can be seen in Figure 3.6.

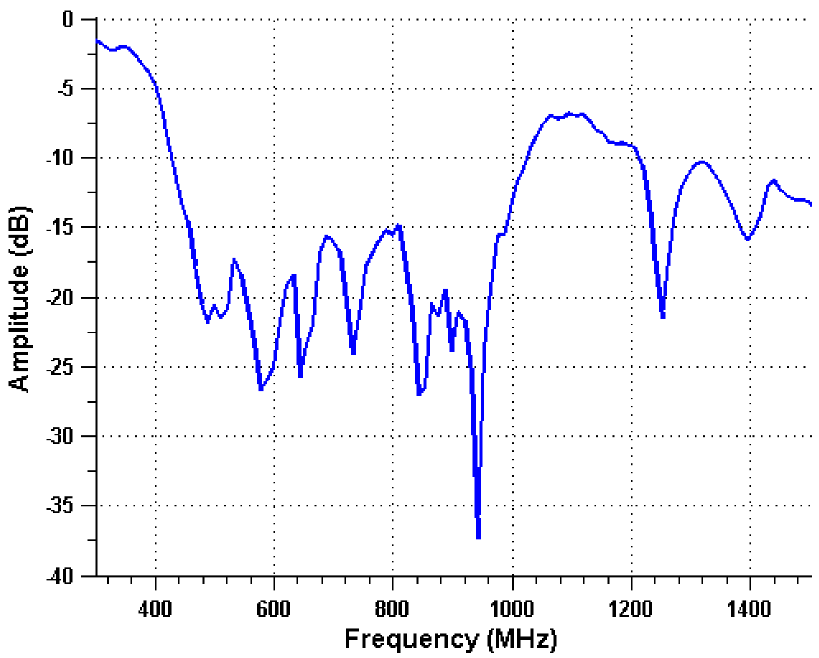


Figure 3.5. Measured reflection curve (S11) of LPA used in the AFIT RNR.

The second change to the model is a subsystem that simulates system noise. Although there are many sources of noise, it is often assumed that the internal noise of the receiver is the dominant contributor to the system noise, and all other sources are

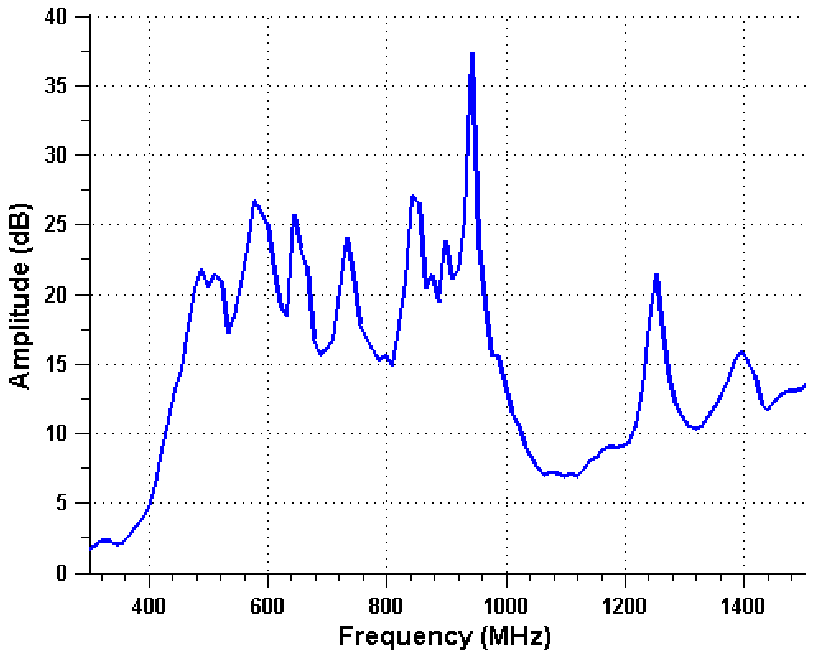


Figure 3.6. The inverse of the S11 curve is assumed to be proportional to the gain of the LPA.

assumed to be zero. Internal noise is considered to be white Gaussian, thermal noise and is proportional to the bandwidth, B, of the receiver and given by equation (2.4). Since system noise is added to the receive signal and cannot be separated from the signal, any amplifications to the signal throughout the receiver also amplify the noise.

Finally, the target environment subsystem must accurately reflect the expected behavior of the AFIT RNR. This subsystem must account for the spreading loss of the signal due to the antenna, the radar cross section of the target, the range from the radar to the target, and the measurement window (integration time) of the RNR.

The environment subsystem is much more difficult to model than the AFIT RNR system components. It is even more difficult to validate. The simplified environment subsystem models the radar return only from a single target of a specific geometry at a specific distance from the radar. The model does not account for external electromagnetic interference, antenna polarization and interference, and multi-bounce echoes from surrounding walls, floors, and ceilings. Those external influences are beyond the scope of this effort and not included in the model, but are expected to cause the most significant discrepancies between the measured and simulated data.

3.5 Performance Experiment

In order to determine the accuracy of the Simulink® model, a comparison of the model results to measured data is warranted. This section discusses the test equipment and setup, as well as the procedures developed to gather relevant data.

3.5.1 Equipment and Setup.

To collect the data used to compare to the model, a single node of the AFIT RNR is used in its near-monostatic configuration. Along with the RNR node, a Tektronix DPO 7354 Digital Phosphor Oscilloscope is used to gather voltage and spectrum information. A single, 4 ft^2 flat-plate target is placed at various distances from the radar. The radar cross section (RCS) for this specific target can be easily modeled in the RNR Simulink® model. The AFIT Compact Electromagnetic RCS Range (ACER) was chosen as the test location because of its size, and because it is designed to minimize external electromagnetic interference. Unfortunately, the ACER range is not optimized for the long wavelengths of the UHF band that makes up the AFIT RNR transmit signal. Multi-bounce echoes and returns from the walls, ceilings, and floors are expected. The RNR along with the collection equipment and setup can be seen in Figure 3.7. A view of the flat-plate target at a distance of 8 m from the RNR can be seen in Figure 3.8.

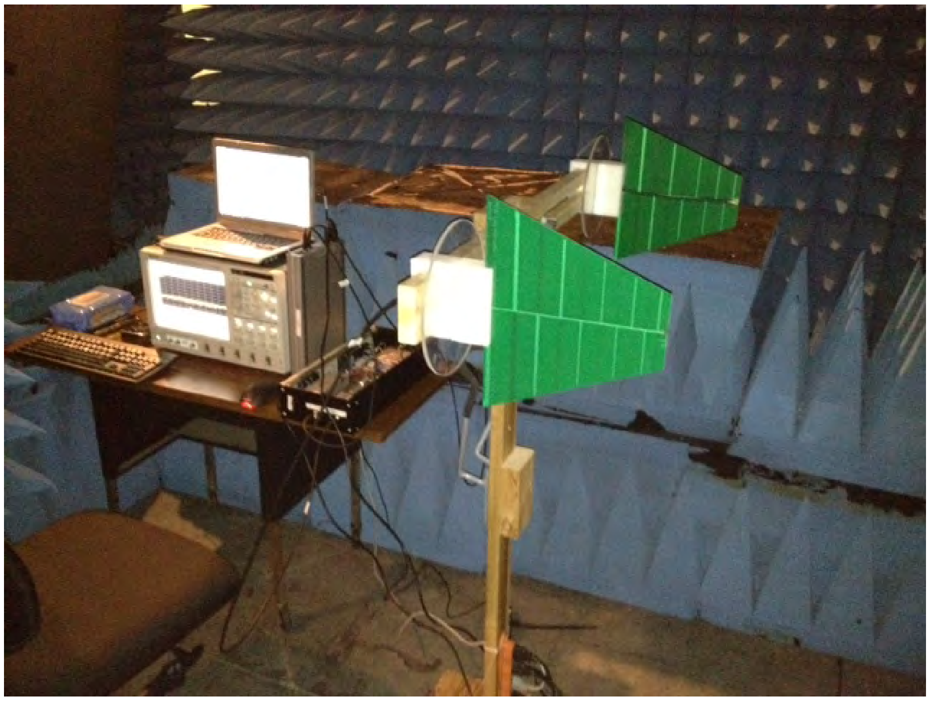


Figure 3.7. A single node of the AFIT RNR is shown along with the collection equipment used in the performance experiment.

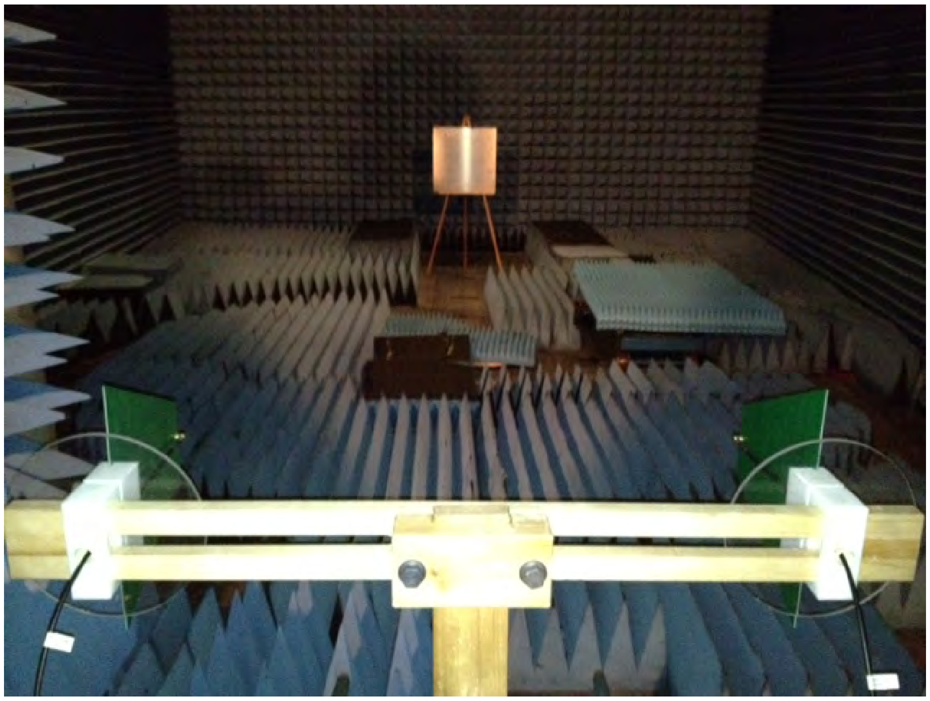


Figure 3.8. The 2 ft^2 flat-plate target located at a distance of 8 m from the RNR is shown.

3.5.2 Experiment Procedures.

Comparing a single measured result to the model will not provide much confidence in the robustness of the model. To truly determine the accuracy of the model, multiple measurements in multiple configurations must be compared to the model. However, due to time, equipment, and location limitations, the number of radar and target configurations is limited. Therefore, receive path measurements are taken for a single target at 6 and 8 m using HH and VV antenna polarization, and using two different RNRs. The target chosen is a flat plat with a hip-pocket radar cross section (σ) equation given by [1]

$$\sigma = \frac{4\pi w^2 h^2}{\lambda^2}, \tag{3.5}$$

where w and h represent the width and height of the flat plate in meters. The flat plate chosen for this experiment is a square reflective plate with 2 ft sides.

The measurements are taken at five test points which can be seen in Figure 3.9. The first three test points are on the transmit path, where measurements are taken for both RNR units at the output of the noise source (TP1), the output of the filters (TP2), and the output of the splitter (TP3). The last two test points occur on the receive path right after the receive antenna (TP4), as well as right before the ADC (TP5) to determine the impact the receiver has on the received signal. To ensure consistency in the noise signal measurements, 100 realizations of the signal at each test point will be averaged. The planned receive path measurements can be seen in Table 3.3.

Not only are transmit and receive signal measurements collected during the experiment, but the RNR's correlation results are also recorded for targets at 6 and 8 meters, and with the antennas in HH and VV polarization. Similarly, correlation

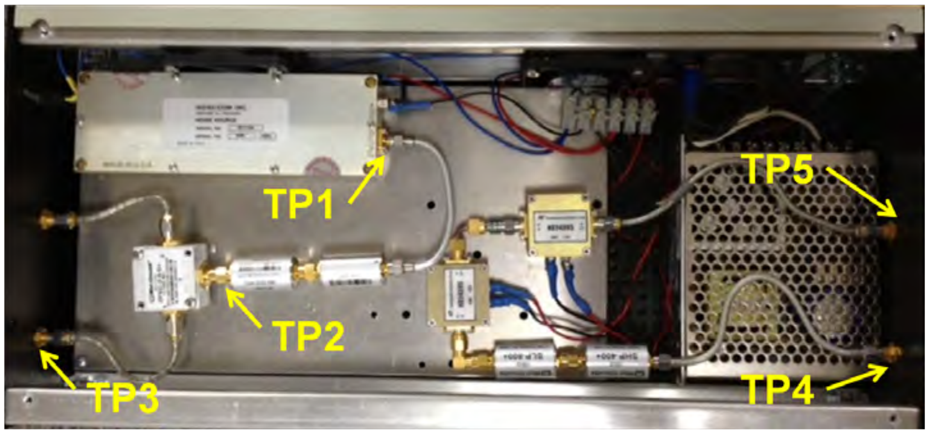


Figure 3.9. The five measurement test points are highlighted on a picture of the AFIT RNR hardware box.

Table 3.3. Pre-Processing SNR Characterization Experimental Test Measurements for 1 μs measurement window on receive path.

Collection Number	Measurement Location	Target Range	Polarization	RNR Box Number
101	Rx Antenna (TP4)	7 m	HH	5
102	ADC (TP5)	7 m	HH	5
103	Rx Antenna (TP4)	7 m	HH	3
104	ADC (TP5)	7 m	HH	3
105	Rx Antenna (TP4)	7 m	VV	5
106	ADC (TP5)	7 m	VV	5
107	Rx Antenna (TP4)	7 m	VV	3
108	ADC (TP5)	7 m	VV	3
109	Rx Antenna (TP4)	5 m	HH	5
110	ADC (TP5)	5 m	HH	5
111	Rx Antenna (TP4)	5 m	HH	3
112	ADC (TP5)	5 m	HH	3
113	Rx Antenna (TP4)	5 m	VV	5
114	ADC (TP5)	5 m	VV	5
115	Rx Antenna (TP4)	5 m	VV	3
116	ADC (TP5)	5 m	VV	3

results of the environment with no target present are recorded, and used for background subtraction. The peak-to-average sidelobe ratio is then determined based on the correlation results after background subtraction. As stated in Section 2.5.4, the peak-to-average sidelobe ratio can be considered the post-processing SNR, and is expected to be approximately equal to the number of samples in the signal, N.

The measurements described above and in Table 3.3 are repeated using two EMCO 3106 UHF standard gain horns in place of the LPAs. These standard gain horns are designed to provide fairly consistent gain in the UHF band, particularly in the AFIT RNR frequencies of interest. Unlike the LPAs, the antenna gain as a function of frequency has been measured and is known for these two horns. The frequency response of the horns can be seen in Figure 3.10. With a known frequency response, the standard gain horns provide an opportunity to verify the model's accuracy by eliminating an unknown variable introduced by the LPAs. The experimental setup within the ACER test range using the standard gain horns can be seen in Figures 3.11 and 3.12.

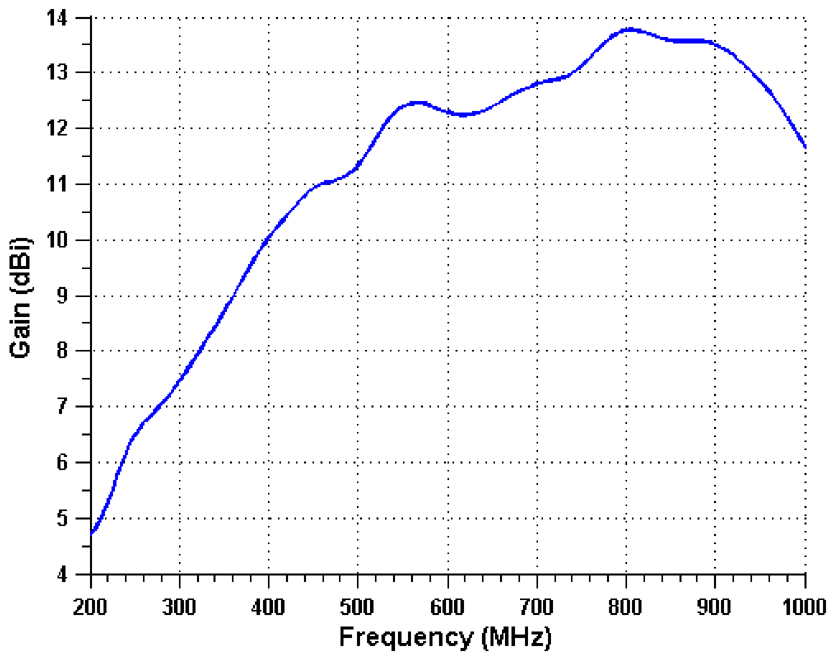


Figure 3.10. The standard gain horn's gain as a function of frequency is known, eliminating an unknown variable in the Simulink® model.

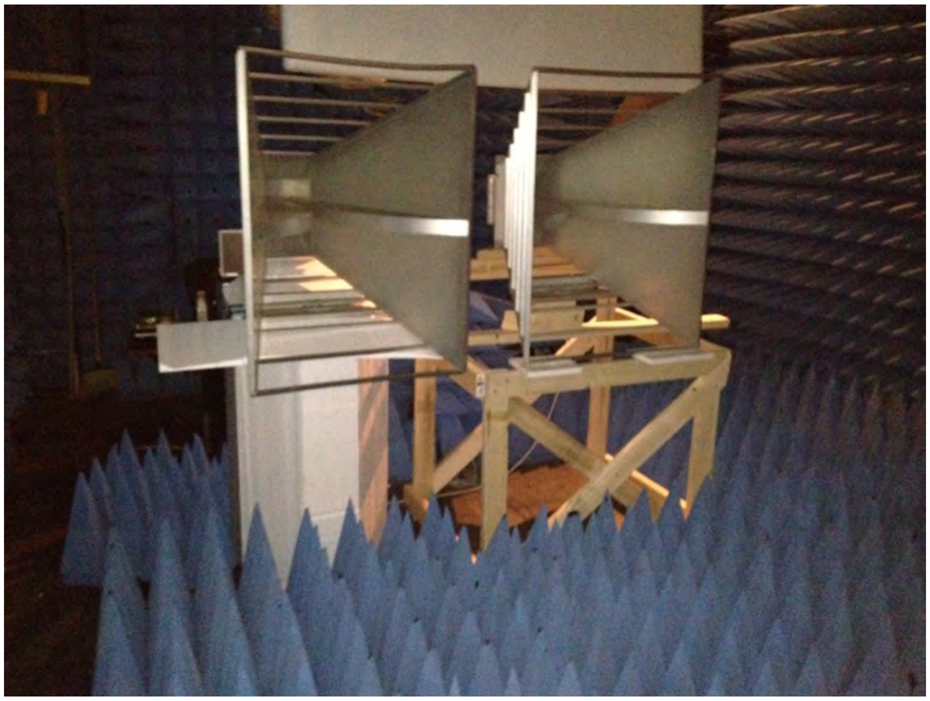


Figure 3.11. Two UHF standard gain horns are used in place of the LPAs to eliminate an unknown variable in the AFIT RNR model.

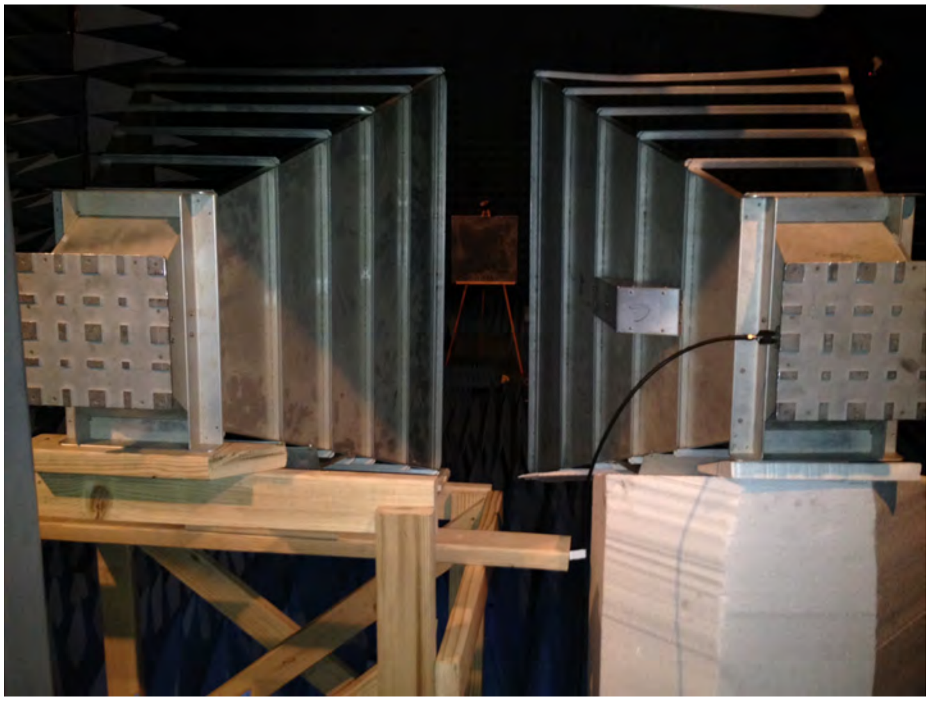


Figure 3.12. The 2 ft^2 flat-plate target located at a distance of 6 m from the RNR with two standard gain horns is shown.

3.6 Chapter Conclusion

This chapter detailed the methodology for improving the simultaneous range and velocity processing in the AFIT RNR. It presented a logical approach to minimizing the memory requirement and configuring the 2D processing algorithm for implementation on a GPU. Furthermore, this chapter presented the structure of the AFIT RNR Simulink® model and the construct for collecting physical measurements of the AFIT RNR to compare to the model. The results of the 2D processing optimization effort and the model comparison are presented in the next chapter.

IV. Results

4.1 Chapter Overview

The results of the simulations and experiments discussed in the previous chapter are presented here. Comparisons are made between the theoretical, simulated and/or measured results, providing a comprehensive analysis of the primary research objectives. The chapter begins a summary of the 2D processing experimental effort, followed by a graphical comparison of the simulated and measured signals within the AFIT RNR. Additionally, a numerical comparison of the calculated, simulated and measured SNR results is presented.

4.2 2D Processing Results

Improvements to the AFIT RNR simultaneous range and velocity processing algorithm were made in a systematic and logical progression. The first effort focused on replacing the eight-bit ADC with a binary ADC. The second effort involved modifying the algorithm for parallelization on a GPU. This section presents the results of these two efforts.

4.2.1 Binary ADC Results.

The results of the eight-step test discussed in Section 3.3.1 are summarized in Table 4.1. The procedure began by recreating the successful results that Lievsay published in [18]. The processing time was $T_p = 42$ minutes on the fastest computer used by Lievsay. This test, however, used a newer computer and the processing was only $T_p = 37.5$ minutes using his data and algorithm. After optimizing Lievsay's algorithm by eliminating superfluous variables and changing the transmit and receive signal variables from double to single precision in MATLAB®, the processing time improved to $T_p = 15.5$ minutes. As expected, no impact to the accuracy of the results were found when using single precision.

To simulate the output of a binary ADC, only the sign of the transmit and receive variables was used for the next step. As hypothesized, the results were consistent with Lievsay's [18]. Although the memory requirements and processing times were improved to this point, the processing time was still prohibitively long, so further optimization was required.

Table 4.1. Summary of binary ADC test results

Step	Description	Peak Memory (GB)	Time (minutes)	Consistent?
1	Lievsay's results [18]	40	37.5	Yes
2	Minor mods	28	22.3	Yes
3	Single precision	21	15.5	Yes
4	Signs Only	21	15.5	Yes
5	Replaced *interp1*	15.5	9.35	Yes
6	Parallel	42	5.38	Yes
7	FFT points	35	4.7	Yes*
8	GPU feasibility	–	–	–

* Slight decrease to velocity resolution

MATLAB®'s *interp1* function was used by Lievsay [18] to generate a bank of reference signals and was the most time consuming line in the code, necessitating an alternative. There are many features built into the *interp1* function that make the function versatile for many applications, but also slow it's processing speed. For the purposes of this algorithm, a simple index shift to the transmitted signal based on the target's relative radial velocity is all that is needed to create the bank of reference signals. To understand the index shifts required, consider the time signal in Figure 4.1. The simulated transmit signal is plotted with a dotted line along with the simulated receive signal in the dashed line. The receive signal measurement window is shortened at the first sample index by Δt, at the second sample index by $2\Delta t$ and so on. The received signal is compressed as a result of the target's relative radial velocity as detailed in (2.23); however, the received signal is still sampled at the same sample time as the transmitted signal, T_s. The number of index shifts expected in the receive signal is equal to

$$L = \left\lceil \frac{N|\Delta t|}{T_s} \right\rceil, \tag{4.1}$$

where L is the number of shifts, N is the number of samples in the signal, and $\lceil \cdot \rceil$ represents the ceiling, or rounding of a non-integer up to the next integer. For an inbound target, the first $\lceil T_s/(2|\Delta t|) \rceil$ samples of the reference signal will be mapped to the same sample values as the transmitted signal. The next $\lfloor T_s/|\Delta t| - T_s \rfloor$ samples of the reference signal, where $\lfloor \cdot \rfloor$ represents a round-down to the next integer, will instead be mapped to the respective transmitted signal values plus one sample. Similarly, the next $\lfloor T_s/|\Delta t| - T_s \rfloor$ are mapped to the values corresponding to the same samples plus two samples and so on. For a target with unknown velocity, a bank of reference signals are built using this approach to compare to the received signal to the reference signal. The reference signal built with the reference velocity corresponding to the target's actual velocity has a maximum correlation with the received signal.

After replacing the *interp1* function with the algorithm discussed above, the memory usage of the correlation routine was sufficiently low to use MATLAB®'s parallel processing toolbox. Running the routine in parallel across four processors decreased the processing time to 5.38 minutes while maintaining the same results. Next, in step seven, the *fft* function used for signal correlation was optimized to improve processing speed and memory. MATLAB®'s *fft* function operates most efficiently when the number of points in the FFT is set to a power of two. By default, the number of points used is equal to the length of the vector being transformed. Lievsay [18] kept the default and the FFT was running N_{FFT} = 200M points. To speed up the FFT, the number of points was set to the nearest power of two without exceeding

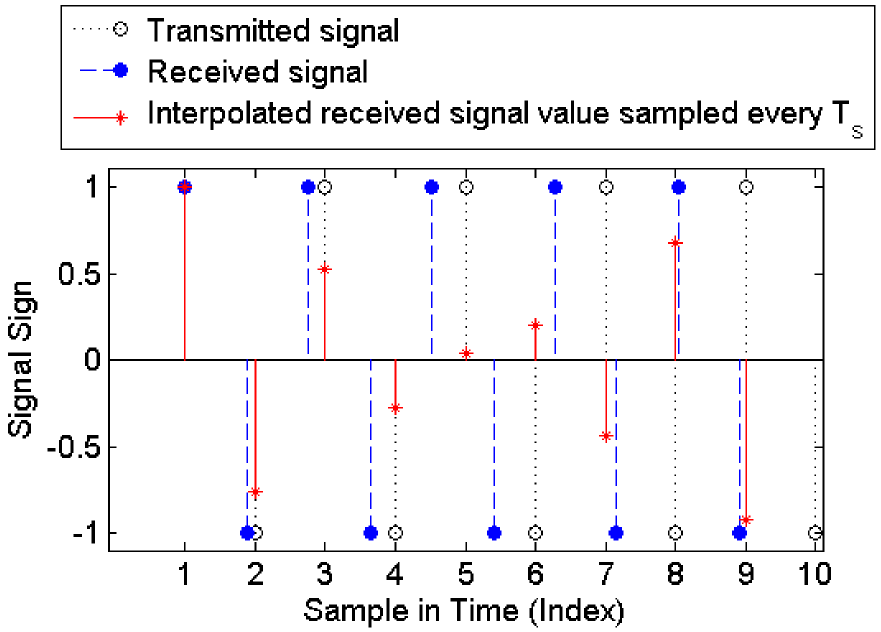
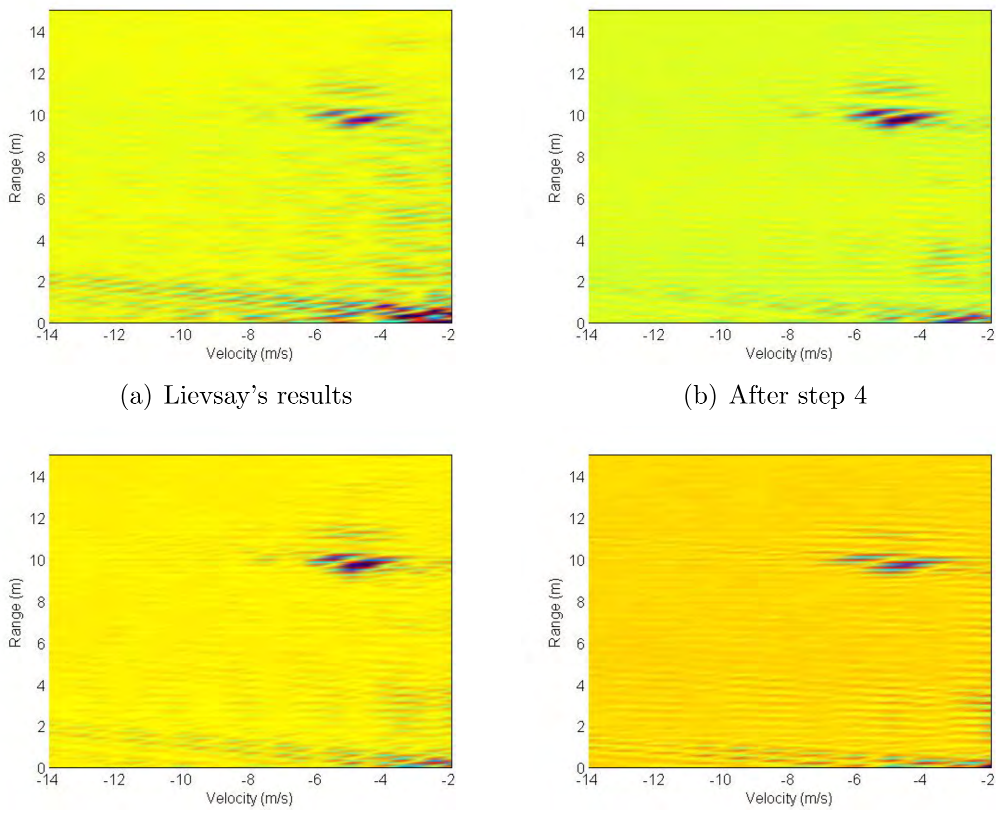


Figure 4.1. Representation of received signal compared to transmitted signal for an inbound target.

$N_{FFT} = 200\text{M}$, which is $2^{27} = 134217728$. The result of this change to the FFT was a 7 GB reduction in RAM and over half a minute improvement to parallel processing using four cores of the computer. However, the drawback to this approach is the elongating of the target response along the velocity axis as can be seen in Figure 4.2. The elongation that results from fewer FFT points is equivalent to diminished velocity resolution due to a shorter measurement window, T. For this target scenario, however, the diminished velocity resolution is negligible.

Not only did the range and velocity estimation performance remain the same using only the signs of the received and reference signals for correlation, but as expected, the post-processing SNR did not change significantly. As discussed in Section 2.5.4, the pre-processing SNR has an impact on the post-processing SNR. Quantization noise affects the pre-processing SNR as seen in (2.29), limiting the maximum pre-processing SNR in a binary ADC to 7.86 dB. For a dominant target with a pre-processing SNR



(a) Lievsay's results (b) After step 4

(c) After step 6 (d) After step 7

Figure 4.2. Results for target at 10 m with inbound relative radial velocity of 5 m/s.

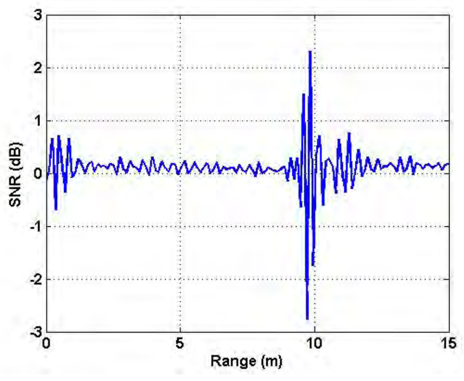
greater than 7.86 dB, the binary ADC limitations will have a negligible impact to the post processing SNR defined by (2.36). In fact, the post-processing SNR actually improved slightly as can be seen in Table 4.2. Only a subset of the eight steps is shown in this table, but the results show no negative impact to the post-processing SNR by using a binary ADC compared to an eight-bit ADC for a single target using the AFIT RNR.

A graphical view of the post-processing SNR comparison can be seen in the range-velocity plots in Figures 4.3 and 4.4. In Figure 4.3, the post-processing SNR for the full eight-bit data is plotted against (a) range perfectly matched in velocity and (b) velocity perfectly matched in range. In Figure 4.4, the post-processing SNR for the simulated binary data is plotted in two-dimensions as well. A comparison of these

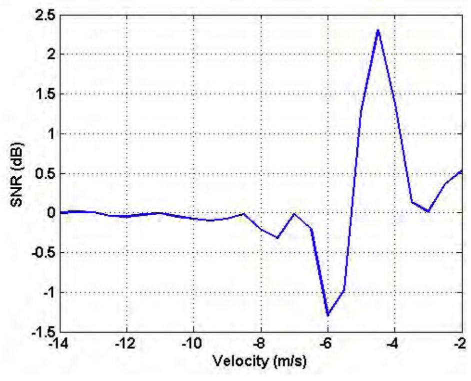
Table 4.2. Post-processing SNR comparison of 2D processing results.

Step	Description	ADC type (simulated)	Change in SNR
1	Lievsay's results	Eight-bit ADC	-
4	Signs only	Binary ADC (sim)	0.5 dB
5	Replaced *interp1*	Binary ADC (sim)	1.3 dB
7	FFT points	Binary ADC (sim)	1.1 dB

figures shows that the binary ADC will not degrade post-processing SNR for a single target with sufficient pre-processing SNR.

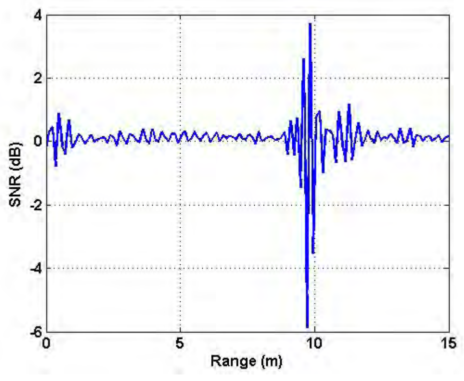


(a)

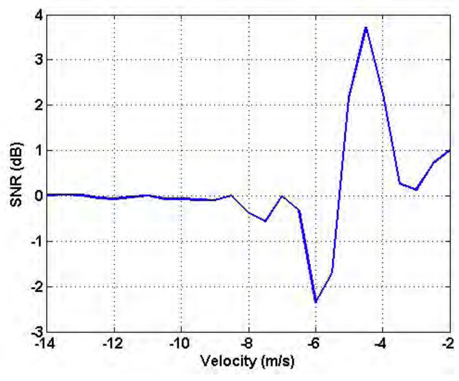


(b)

Figure 4.3. Full eight-bit ADC post-processing SNR plotted against (a) range perfectly matched in velocity and (b) velocity perfectly matched in range.



(a)



(b)

Figure 4.4. Simulated binary ADC post-processing SNR plotted against (a) range perfectly matched in velocity and (b) velocity perfectly matched in range.

At this point, it is clear that switching to a binary ADC alone does not allow for mass parallelization in the AFIT RNR using a GPU or FPGA. Further memory optimization must be accomplished to take advantage of the available GPU. The next section details the results of such an optimization effort aimed at reducing the memory burden in order to perform correlation processing on a GPU.

4.2.2 FFT Segmentation Results.

The second phase of the 2D processing improvements involved the three-step process discussed in Section 3.3.2. After configuring the AFIT RNR 2D processing algorithm for implementation on the GPU as previously discussed, the algorithm was applied to the sample transmit and receive data according to the test procedure. As can be seen in Table 3.2, the two computers used for the 2D processing share similar specifications. The first computer, however, has faster processing speeds leading to faster CPU operations. The second computer is outfitted with a higher-end GPU model and resulted in faster GPU operations. This difference in processing hardware explains why, when comparing Table 4.3 to Table 4.4, the first computer performs the 2D processing faster when only the local CPUs are used. Conversely, the second computer performs the 2D processing faster when the GPU is tasked with the majority of the signal processing.

Table 4.3. Summary of computer 1 processing times per reference velocity

Step	Description	Reference Signal Time (seconds)	Correlation Time (seconds)	Total Time (seconds)
1	No GPU	1.49	8.27	9.76
2	MATLAB®	1.55	3.72	5.27
3	Jacket®	1.92	9.42	11.34

Another factor affecting the speed of the 2D processing is the GPU interface used. Table 4.5 reveals that the MATLAB® GPU implementation is faster than Jacket®

Table 4.4. Summary of computer 2 processing times per reference velocity

Step	Description	Reference Signal Time (seconds)	Correlation Time (seconds)	Total Time (seconds)
1	No GPU	1.78	10.23	12.01
2	MATLAB®	1.70	2.61	4.31
3	Jacket®	1.98	8.73	10.71

for this 2D processing algorithm. MATLAB®'s parallel computing toolbox performs faster because of its inherent speed in matrix math. The FFT segments were placed in matrix format to process many FFTs simultaneously, thus reducing the number of loops required to process all FFT segments for correlation. Jacket®, on the other hand, has a very efficient *gfor* loop not available on the MATLAB® interface, but its known memory allocation issues did not allow for efficient matrix FFT calculations.

Table 4.5. Summary of 2D processing time for 25 reference velocities

Step	Description	Computer 1 Time (minutes)	Computer 2 Time (minutes)
1	No GPU	4.07	5.00
2	MATLAB®	2.20	1.80
3	Jacket®	4.75	4.46

Tables 4.3 and 4.4 also reveal the fact that reference signal generation accounts for approximately 25 to 30% of the overall 2D processing time. This signal generation time can be eliminated by using a template playback in place of the thermal noise source. A template playback strategy involves building a digital noise transmit signal and using a digital-to-analog converter (DAC) in place of the thermal noise generator. This template playback approach introduces periodicity and leads to a transmit signal that is not truly random, but the desired low probability of intercept attribute can be preserved using strategies as discussed in [30].

Generating the reference signal bank *a priori* would significantly reduce the 2D processing time of the AFIT RNR, but it would require a significant amount of memory to store all the reference signals. The current algorithm steps through a vector of reference velocities. For each reference velocity, it builds a single reference signal and performs the cross-correlation with the receive signal. Once the cross correlation is complete and stored, the algorithm clears the reference signal from memory and generates a new reference signal based on the next reference velocity. This iterative process takes time but minimizes the number of lengthy signals that must be stored in memory.

Although the reference signal generation time is a major factor in the overall processing time, the correlation processing consumes the majority of the overall time and is still prohibitively long. Improvements to the correlation algorithm need to be made for this algorithm to be applied in a practical radar application. Some thoughts for future work will be discussed in the next chapter.

4.2.3 2D Processing Analysis.

Decreasing the simultaneous range-velocity processing time of the AFIT RNR by more than an order of magnitude from $T_p \approx 42$ minutes to $T_p < 2$ minutes is a significant improvement. Unfortunately, a significant effort is still required to bring the processing time to near real-time. Simultaneous range and velocity processing has its advantage in that it can separate two targets traveling at different velocities within the same range bin. This characteristic is highly desired for collision avoidance in autonomous vehicles. Future research efforts in the AFIT RNR should continue to set near-real time 2D processing as a primary objective.

4.3 SNR Analysis Results

This section presents the results of the SNR analysis effort described in the previous chapter. First, a graphical comparison is made between the Simulink® model and the measured data. Second, the results of the AFIT RNR SNR analysis is summarized.

4.3.1 Simulink® Model Results.

A software model is useful to allow the radar designer to explore the current design as well as determine the feasibility of proposed design changes. As the AFIT RNR progresses toward a collision avoidance application, many design changes are on the horizon. A model was developed in Simulink® to emulate the performance of the AFIT RNR. The top layer of the AFIT RNR Simulink® model can be seen in Figure 4.5.

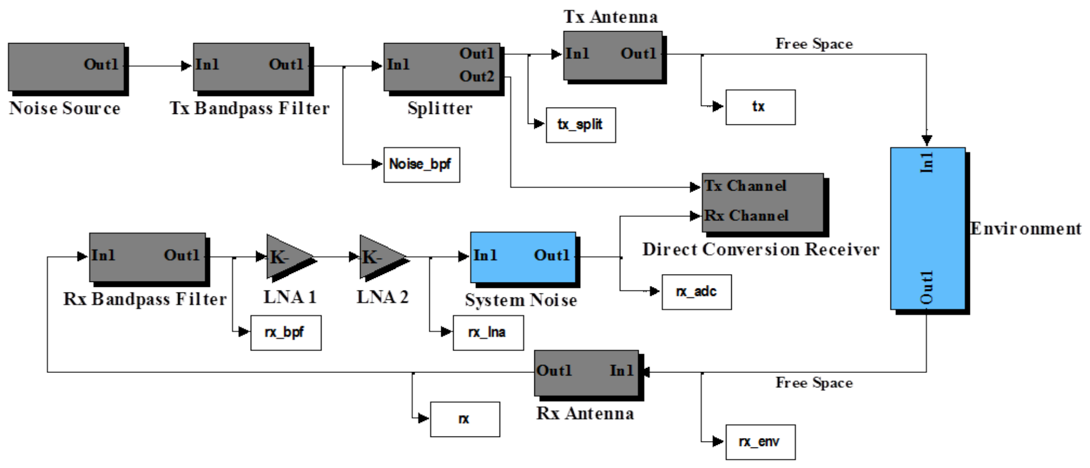


Figure 4.5. The top layer of the AFIT RNR Simulink® model, designed to aid in determining the feasibility of proposed hardware design changes.

To prove the accuracy of the model and its effectiveness as a design tool, the simulated performance must be compared to measured performance. Measurements were taken according to the procedures outlined in Section 3.5.2 and are presented here and in Appendix A. There is, however, negligible difference between measured results of the two AFIT RNR hardware boxes. Because of the similarities, only the results from AFIT RNR hardware box 3 are presented in this chapter. However, all results can be seen in Appendix A.

The model must be able to approximate the transmit signal effectively in order to match measured AFIT SNR results to be discussed in Section 4.3.2. As discussed in Section 3.2.1, the transmit path includes the thermal noise source, an LPF, a HPF, and a signal splitter. Those components are directly modeled in Simulink® as seen in Figure 4.6.

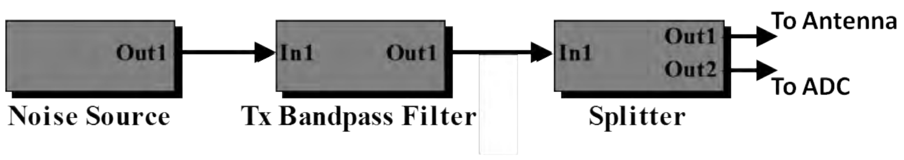


Figure 4.6. Components of the AFIT RNR transmit path as modeled in Simulink®.

The results of the simulated transmit signal power spectral density compared to the average measured signal power spectral density can be seen in Figure 4.7, where it is clear that the transmit path is modeled accurately in the AFIT RNR Simulink® model. The noise signal in black represents a single realization of the simulated, or expected, signal resulting from the thermal noise source. The signal in blue is a single realization of the simulated transmit signal after band-pass filtering using the LPF and HPF. The green dotted line represents the measured noise signal emanating from the thermal noise generator. As with all measured signals to be presented, the measured signal was averaged across 100 realizations to ensure any abnormalities or spurious interferences are minimized. Finally, the red dotted line represents the transmit signal measured immediately after the band-pass filtering. A single realization of the modeled results is presented in Figure 4.7 to highlight the variance within the signal due to the WGN source. In the remaining graphical comparisons presented in this chapter and in Appendix A, the model results will be averaged across 100 realizations for a direct comparison with the measured results.

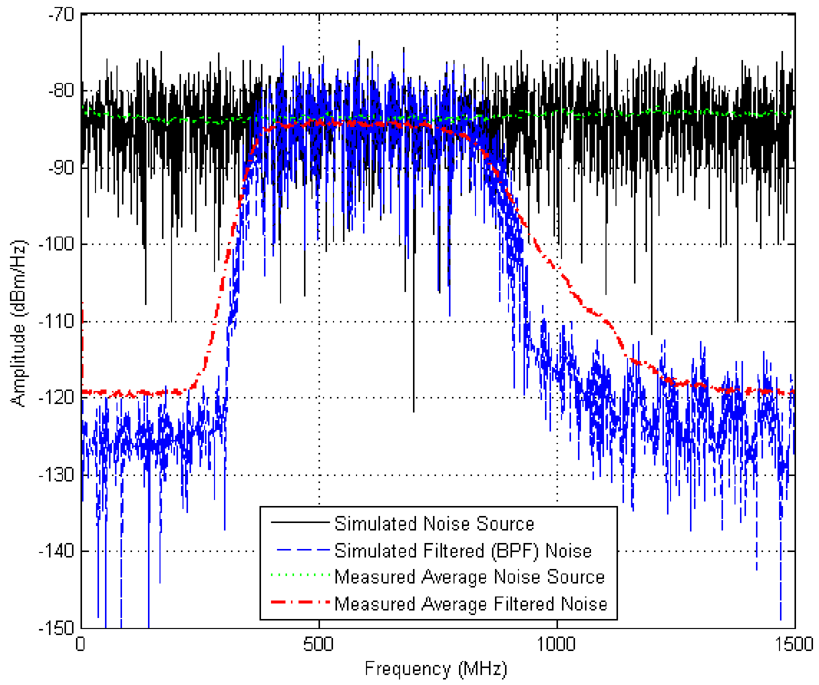


Figure 4.7. Comparison of the average measured transmit signal with a single realization of the transmit signal modeled in Simulink®.

The environment subsystem is designed to model the return from a single, stationary target in an unknown environment. It was constructed using three gain blocks and a transport delay block as can be seen in Figure 4.8. The first gain block labeled “PathLoss1” represents the range to the target and the spreading loss, given by $1/((4\pi)^3 R^4)$. The “RCS1” gain block represents the target gain factor. For the flat-plate target used in this research effort, the target gain factor is a combination of the target RCS, the wavelength squared and the time-bandwidth product given by $4\pi w^2 h^2 \lambda^2 BT/\lambda^2$. The range to a target is determined in a radar based on the two-way transit time which is represented in the environment subsystem by the transport delay block labeled “Delay1”.

The final gain block in the environment subsystem is labeled "EnvLoss" and is essentially a "catch-all" to account for external environmental factors. Multipath, antenna coupling, antenna polarization, clutter, and external interference are just a few of the factors that affect the return of a radar signal from a real-world environment.

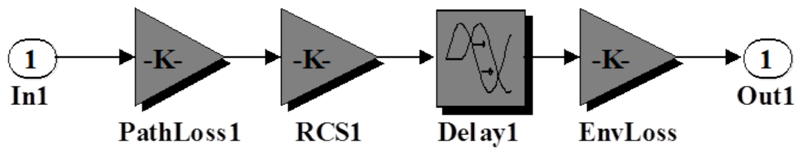
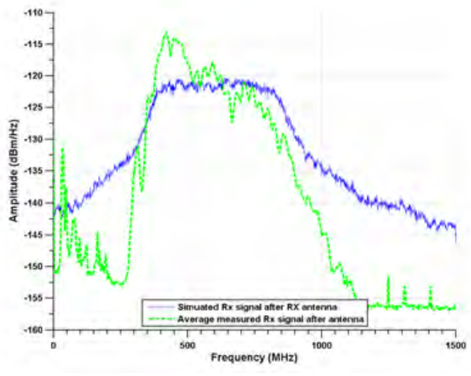


Figure 4.8. The environment subsystem, used to model the return from a single stationary target, is composed of three gain blocks and a transport delay block.

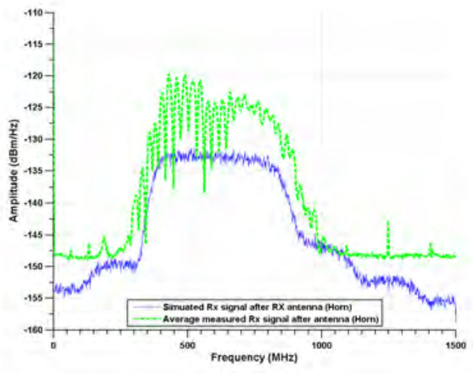
To determine the accuracy of the environment subsystem of the AFIT RNR Simulink® model, a comparison between the measured receive signal and the modeled signal is required. As stated in Section 3.4.2, the environment subsystem was expected to cause the most significant discrepancy between the measured and simulated data. That expectation has proven to be true as can be seen in Figure 4.9, where the measured and simulated receive signal from a target at 6 m using VV polarization is illustrated. A comparison of the measured receive signal to the simulated receive signal shows that there is some coloring to the receive signal resulting from the external interference discussed in Section 3.4.2.

Even without a detailed environment subsystem, the model effectively estimates the return echo in a general sense. For an example of the model's flexibility, consider Figure 4.10. The return echo has a smaller amplitude than the echo from Figure 4.9 because the target is farther from the radar.

A measurement was taken with the RNR transmitter off and a 50-ohm load on the receive antenna input to analyze the receiver system noise. The results of the

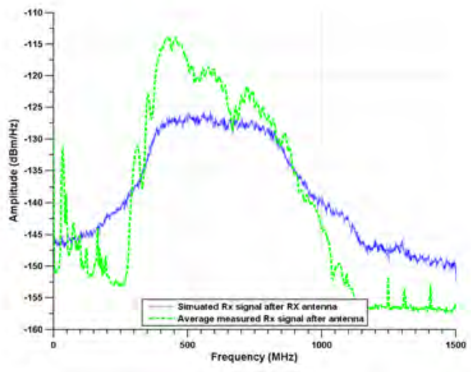


(a)

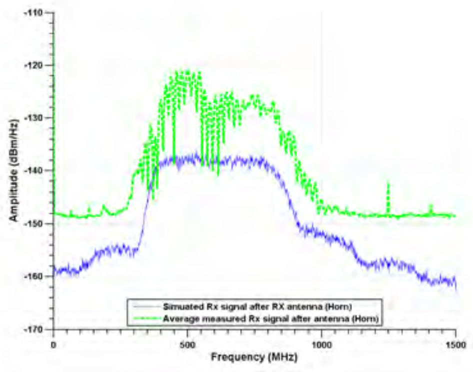


(b)

Figure 4.9. Comparison of the average measured receive signal of target at 6 m in VV polarization with the same signal modeled in Simulink® for (a) the standard AFIT RNR antenna configuration, and (b) the standard gain horns in place of the LPAs.



(a)



(b)

Figure 4.10. Comparison of the average measured receive signal of target at 8 m in VV polarization with the same signal modeled in Simulink® for (a) the standard AFIT RNR antenna configuration, and (b) the standard gain horns in place of the LPAs.

measurements can be seen in Figure 4.11. The noise response has a downward trend across the passband. Because the noise measurement was taken after the cascaded LNAs, a negative slope is expected in the noise measurement. Except for the downward slope caused by the LNAs, the assumption that the received noise power spectral density is not a function of frequency but solely a function of bandwidth as defined by the LPF and HPF is accurate.

Continuing through the receive path, the AFIT RNR Simulink® model accurately

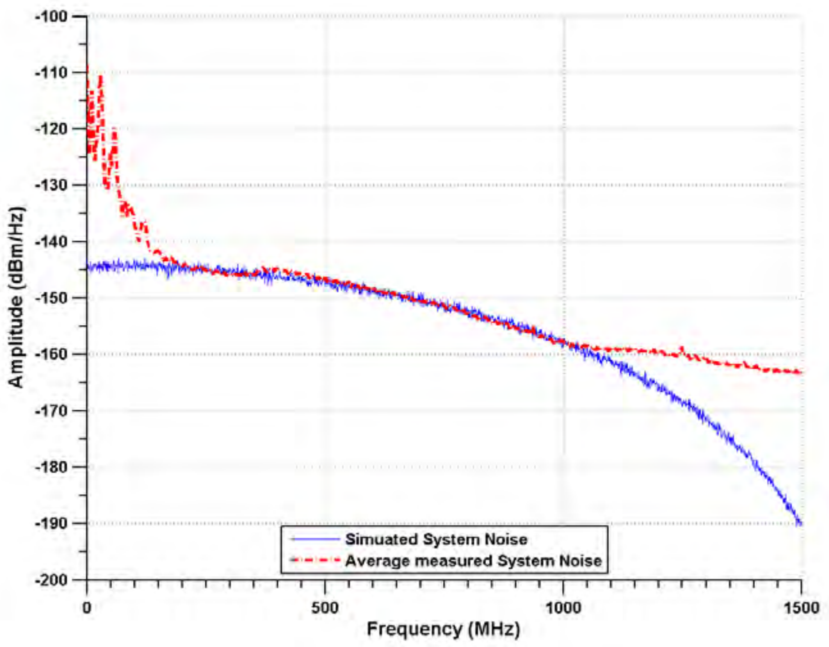
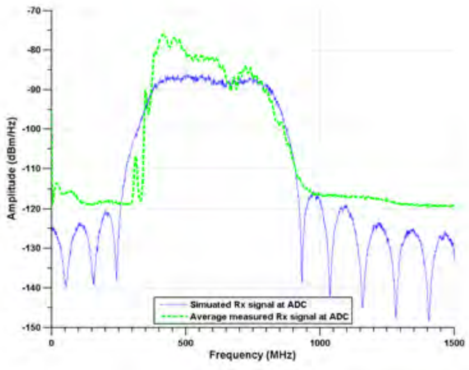


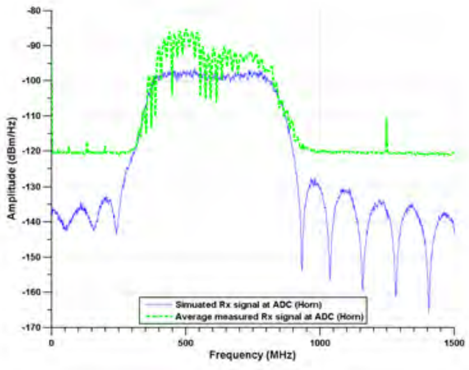
Figure 4.11. The noise floor of the AFIT RNR, measured with the transmitter off and a 50-ohm load on the receive antenna input, compared to the modeled system noise.

predicts the performance of the AFIT RNR receive components. This can be seen by examining Figure 4.12, where the average measured receive signal after the filters and LNAs is compared to the simulated representation of the same signal. The same signal coloring found in Figure 4.10, which is representative of the receive signal before the receive path components, is found in Figure 4.12. Similarly, the signal changes caused by the filters and LNAs are consistent between the measured signal and the simulated version.

The polarization of the antennas also affects the measured signals, as can be seen in Figure 4.13, where a top-down view of the antenna configuration with a basic representation of the antenna pattern is illustrated. The differences between the return echoes of the horizontal (HH) and vertical (VV) polarizations are expected due to the antenna patterns of the LPAs. In the horizontal configuration, the receive antenna will have less cross-talk (or coupling) from the transmit antenna due to the



(a)



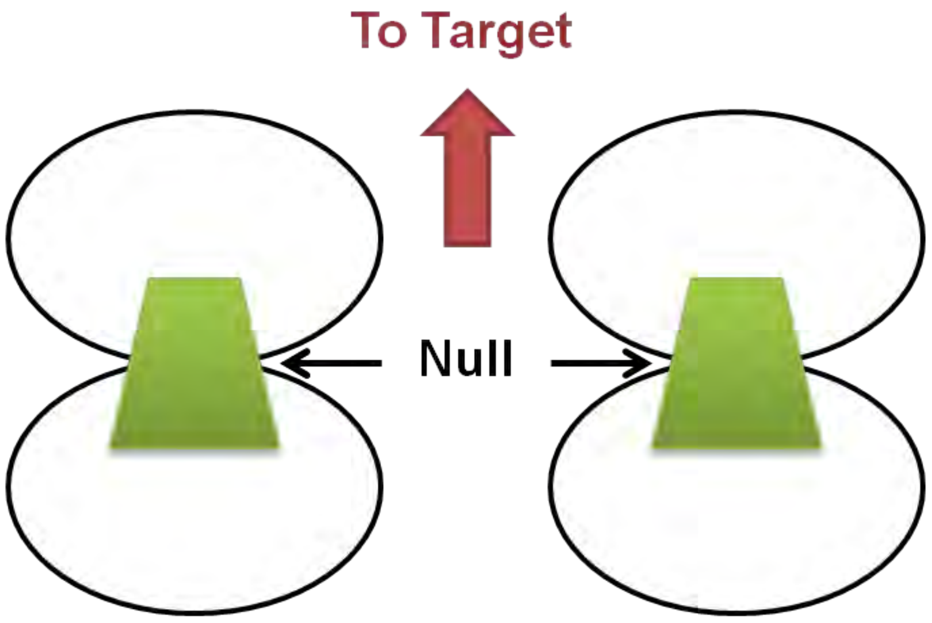
(b)

Figure 4.12. Comparison of the average measured receive signal at the ADC of target at 8 m in VV polarization with the same signal modeled in Simulink® for (a) the standard AFIT RNR antenna configuration, and (b) the standard gain horns in place of the LPAs.

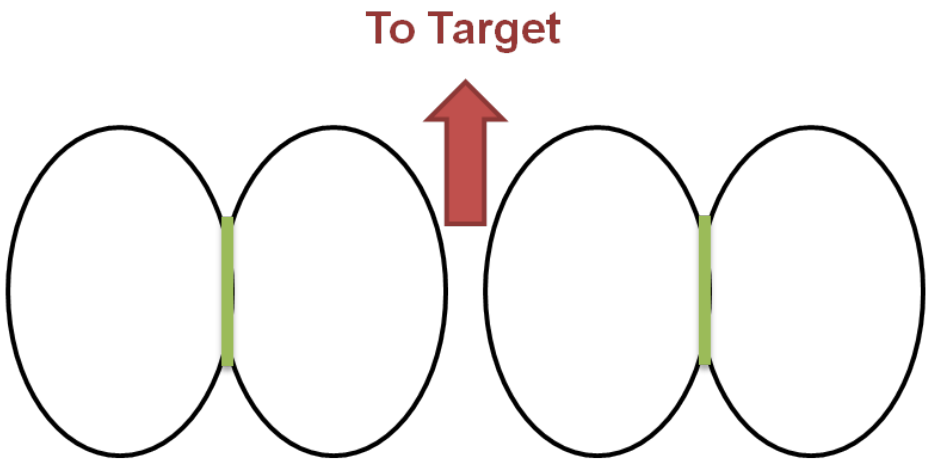
null in the antenna pattern where the receive antenna is located. However, much more energy is returned from the floor in this configuration. With the vertical polarization configuration, the antenna pattern null is pointed at the floor, minimizing returns from the floor. However, there is no null in the direction of the receive antenna, providing significant cross-talk that must be accounted for.

In the Simulink® model, antenna polarization mismatch was simply modeled as a constant change to the environmental loss gain block. This gain factor did not account for the frequency domain signal coloration, but it did generally represent the change in signal amplitude as a result of antenna polarization. Figure 4.14 highlights the overall change in amplitude and coloration difference in HH and VV polarization responses.

The effects of polarization on target detection and estimation can be seen when examining the correlation results. As can be seen in Figure 4.15, the VV polarization cross-talk from the transmit antenna to the receive antenna results in very high correlation at a distance of 1 meter. It is no coincidence that this correlation peak is also the distance between the transmit and receive antennas on the AFIT RNR. In



(a)



(b)

Figure 4.13. In (a) HH configuration, the antenna null reduces coupling from the transmit to receive antenna, but increases the received echo from the floor. The opposite is true for (b) VV polarization.

contrast, the correlation results from measured data in the HH polarization does not offer such significant correlation as a result of antenna cross-talk.

As a final measure of the accuracy of the AFIT RNR model, consider the comparison of the correlation response of the measured data versus the Simulink® model's

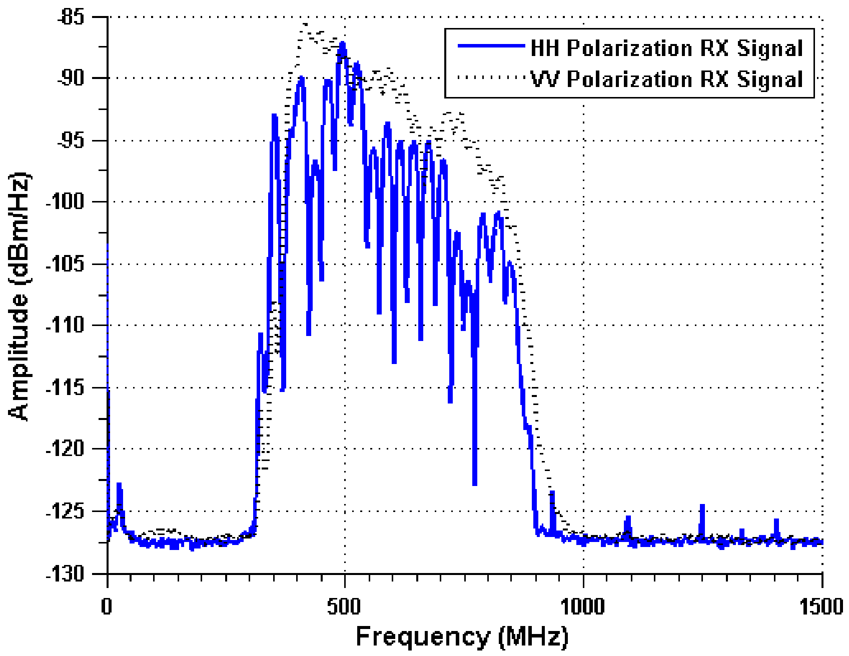


Figure 4.14. Comparison of the average measured receive signal power spectral density at the ADC input in HH and VV polarization using the LPAs.

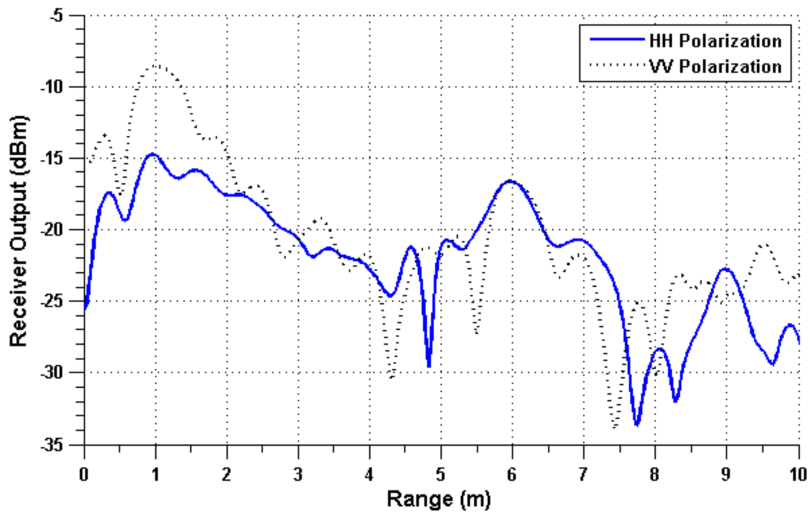
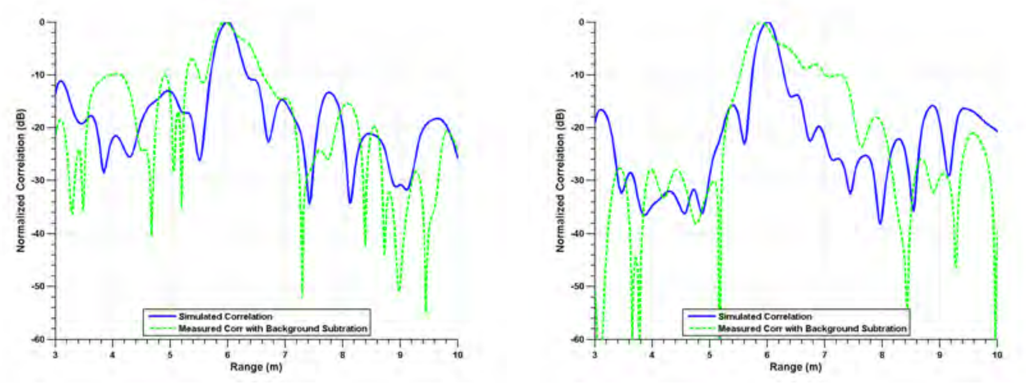


Figure 4.15. Comparison of the measured correlation results of a target at 6 meters using HH and VV polarization.

correlation results as seen in Figure 4.16. The normalized squared magnitude of the correlation results measured for a target at 6 m using HH polarization are compared to the model for the AFIT RNR with its LPAs as well as when the standard gain horn is used in place of the LPAs. The target is clearly identified at 6 m and has a similar post-processing SNR in the simulated and measured results for both the standard and horn-modified AFIT RNR configurations. A thorough discussion of the post-processing SNR is further presented in the next section.



(a) (b)

Figure 4.16. In the (a) standard AFIT RNR configuration with LPAs, the correlation results of a target at 6 meters using HH polarization are consistent. In the (b) horn-modified AFIT RNR configuration, correlation results are also consistent.

4.3.2 SNR Analysis.

There are two basic ways to evaluate the pre-processing SNR of the AFIT RNR. As discussed in section 2.5.2, the signal is a function of frequency, thus the SNR can be characterized as a function of frequency and measured in SNR/Hz. Additionally, the signal and noise power can be integrated across the bandwidth to determine the the total receive signal and noise power. Both methods for examining the AFIT RNR are exploited in this research effort. Included in the SNR analysis is a comparison of the pre-processing SNR calculated using (2.28), the pre-processing SNR resulting from the Simulink® model, and the measured pre-processing SNR. The SNR analysis will culminate in a comparison of the simulated and measured post-processing SNR.

Because the receive signal power is a function of frequency, the pre-processing SNR is consequently a function of frequency and is best viewed on a graph. Figure 4.17 is representative of the SNR results in terms of dB/Hz. The simulated SNR frequency response does not mimic exactly the measured results, but overall the simulated response is close using in the standard AFIT RNR configuration. When the LPAs are replaced with the standard gain horns, the SNR as a function of frequency is not as close when comparing the measured and simulated results. This difference is due to the limitations of the DPO measurement device. The dynamic range of the ADCs in the DPO were adjusted to view the response in the 400 to 800 MHz band of interest, which resulted in ADC saturation and clipping the out-of-band signal. Adjusting the reference level (and hence the dynamic range of the ADCs) to view the out-of-band signal results in clipping of the passband signal. The result is a measurement with an artificially high noise level. More importantly, the frequency response coloration could not be simulated in the model, necessitating a thorough comparison of the integrated SNR in place of this SNR per Hz comparison.

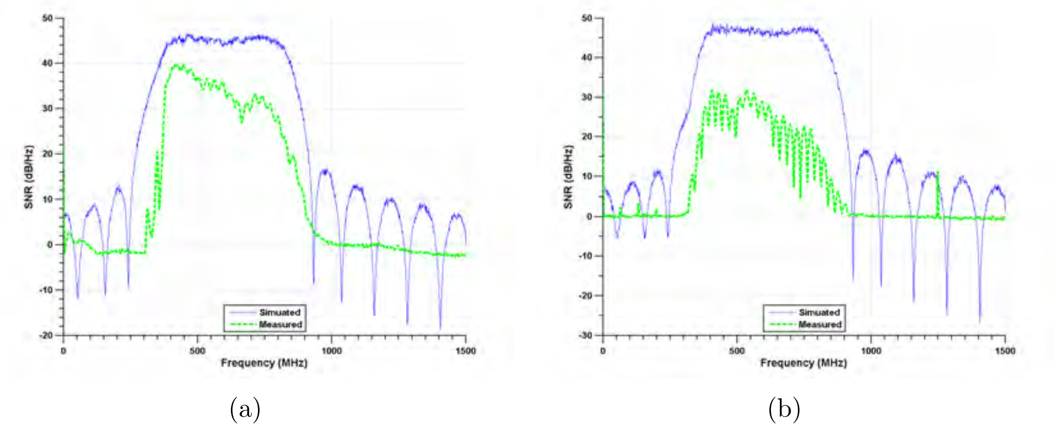


Figure 4.17. A comparison between the simulated and measured pre-processing SNR as a function of frequency is made using (a) the standard AFIT RNR configuration and (b) the horn-modified configuration. Target at 6 meters, HH polarization.

The integrated pre-processing SNR of the AFIT RNR was calculated using (2.28), derived from the Simulink® model results, and derived from the measured data. Ideally, all three SNR results would match exactly. However, as with any other physical system, the results do not match the ideal case exactly. The results, however, are consistent and point to the accuracy of the model. Table 4.6 compares the calculated, simulated, and measured pre-processing SNR results across LPAs in HH and VV polarization, and targets at 6 and 8 meters. Likewise, Table 4.7 compares the calculated, simulated, and measured SNR results across the standard gain horn antennas in HH and VV polarization, and targets at 6 and 8 meters.

Table 4.6. Comparison of calculated vs. simulated vs. measured SNR results

Pol/Range (m)/T (μs)	Calculated SNR (dB)	Simulated SNR (dB)	Measured SNR (dB)
HH/6/1.00	58.489	54.293	49.879
HH/8/1.00	53.492	48.084	50.330
VV/6/1.00	58.489	69.293	65.433
VV/8/1.00	53.492	63.086	65.382

Table 4.7. Comparison of calculated vs. simulated vs. measured pre-processing SNR results using standard gain horns in place of the LPAs

Pol/Range (m)/T (μs)	Calculated SNR (dB)	Simulated SNR (dB)	Measured SNR (dB)
HH/6/1.00	70.489	61.047	60.297
HH/8/1.00	65.492	55.682	59.923
VV/6/1.00	70.489	58.047	57.783
VV/8/1.00	65.492	52.682	55.794

As a whole the results listed in Table 4.6 and Table 4.7 show consistency among the calculated, simulated, and measured pre-processing SNRs, but there are three cases that attribute to inconsistencies that should be highlighted. First, differences between the calculated, simulated, and measured results can be attributed to the antenna gain. For the calculated SNR, a constant antenna gain of 6 dB is used for the LPAs, and 12 dB for the standard gain horns. For the simulated SNR, the estimated gain curve as a function of frequency is used for the LPAs, and the known gain curve as a function of frequency is used for the standard gain horns. These differences in gain lead to inherent differences in the results.

In the second case, the calculated SNR does not change between HH and VV polarization, simply because (2.28) does not account for antenna polarization. As discussed in Section 4.3.1, the antenna polarization was accounted for in the model, leading to higher simulated SNR when using the LPAs in VV polarization versus HH polarization. The measured SNR results also show a higher SNR when using the LPAs in VV polarization. Conversely, higher SNR is measured when using the standard gain horns in HH polarization versus VV polarization. This difference is accounted for in the simulation.

The third inconsistency is found in the measured SNR of a target at 6 versus 8 meters. In both the calculated and simulated SNR results, the target at 8 meters consistently has a lower SNR than the target at 6 meters, as expected. However, the measured SNR results using the LPAs show negligible differences between the flat-plate target at 6 meters and 8 meters. This inconsistency results from the environment in which the measurements were collected. Although the measurements were collected in a radar range with radar absorbing material covering all surfaces, the range was constructed primarily for measurements in the X-band. Because the AFIT RNR operates in the UHF band, the radar absorbing material is less effective than it is for X-band radar. Radar returns from the walls, ceiling, floor, and other equipment in the room are included in the receive signal. The clutter included in the measured receive signal using the LPAs is so high that it reduces the SNR, resulting in no SNR difference between the target at 6 and 8 meters. The differences in measured SNR between the flat-plate target at 6 meters and 8 meters using the standard gain horns, however, is not negligible. In both the HH and VV polarizations, the SNR is lower for the target at 8 meters as expected. The horn antenna pattern is much more directive than the LPAs, resulting in more energy on the target compared the LPAs, making the target more dominant in the environment.

Although the target environment is not ideal, the RNR has no trouble determining target location based on the correlation results. As discussed in Section 2.5.4, the correlation results yield the post-processing SNR, which is expected to be approximately equal to the number of samples in the signal. The number of samples used in the correlation processing for this experiment is $N = 1500$, or 31.7 dB. The measured results, with background subtraction as discussed in Section 3.5.2, and the model simulation results are consistent with this 31.7 dB expectation as seen in Figure 4.18. The simulated post-processing SNR or peak-to-average sidelobe ratio is 29.8 dB, and the post-processing SNR of the measured results is 28.9 dB.

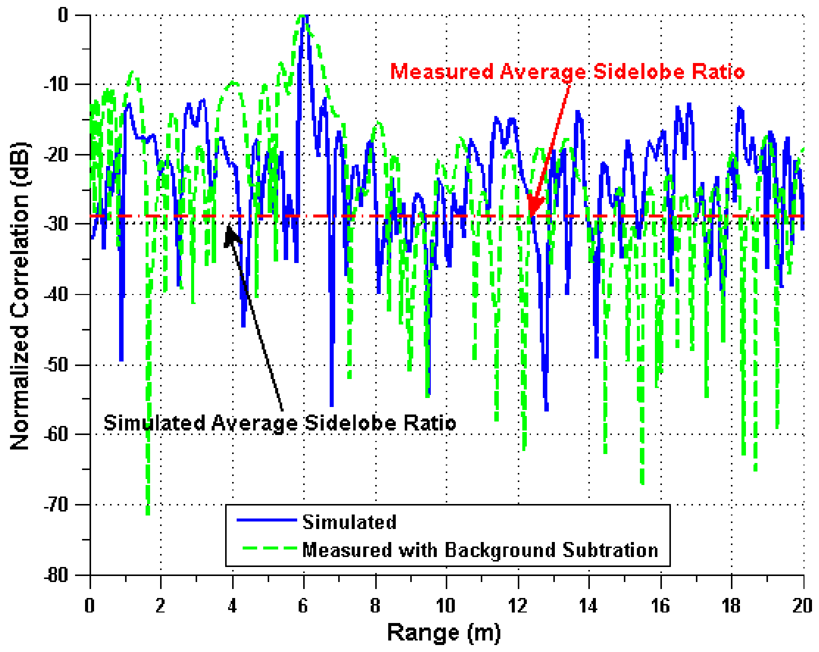


Figure 4.18. This plot presents the normalized results of the simulated and measured correlation with background subtraction. The average sidelobe level is also shown, highlighting the post-processing SNR of the AFIT RNR.

4.4 Chapter Conclusion

The observations presented in this chapter highlight the results of the research effort aimed and evolving the AFIT RNR toward a collision avoidance application. The 2D processing time of the AFIT RNR was reduced over twenty fold, from $T_p \approx 42$ minutes to $T_p < 2$ minutes, and the model developed in Simulink® proved to be an effective representation of the radar system. Additionally, the AFIT RNR was analyzed in terms of the signal-to-noise ratio providing a comparison of the calculated SNR, simulated SNR, and measured SNR. The next chapter will provide a review of the basic research objectives, and draw conclusions based on the research results.

V. Conclusions

5.1 Chapter Overview

The AFIT RNR offers unique sensor characteristics that make its progression to a collision avoidance sensor a logical choice. Before the system can be miniaturized and implemented on a wheeled or airborne autonomous vehicle, a number of improvements must be made. This research effort was aimed at advancing AFIT RNR capability as it progresses towards future applications. This chapter reviews the stated research goals, presents a summary of the research results and contributions, and identifies areas for future study to further advance the AFIT RNR as an effective collision avoidance sensor.

5.2 Research Goals

To be suitable as a collision avoidance sensor on a small autonomous vehicle, the physical hardware size, weight, and power must be minimized. Similarly, the signal processing algorithm must be updated and configured for the application. This research effort focused on the latter, primarily in the areas of range-velocity processing and SNR analysis of the current AFIT RNR. To minimize the 2D processing time, the algorithm was completely overhauled for implementation on a FPGA or GPU. Additionally, the basic software model representing the AFIT RNR required updates to enable efficient analysis supporting future system changes.

5.3 Results and Contributions

The overall research effort resulted in successful achievement of both primary research goals. 2D range-velocity processing was demonstrated using a GPU to distribute the correlation processing across hundreds of processing cores. As a result, the overall processing time for $N_v = 25$ reference velocities was reduced more than an order of magnitude from $T_p \approx 42$ minutes to $T_p < 2$ minutes. Similarly, the research effort demonstrated the feasibility of using a binary ADC in place of the eight-bit ADC to reduce the 2D processing computational burden in high SNR environments. However, the research concluded that real-time 2D processing in the AFIT RNR is not yet feasible using today's technology.

A comprehensive software model was developed to emulate AFIT RNR performance. The model allows the radar engineer to study the effects of hardware changes before actually modifying the system. Additionally, the model enables efficient SNR analysis at all points in the radar system. It was shown that the model accurately predicts system SNR in multiple antenna and target configurations.

5.4 Future Work

Although the research goals were met, considerable effort remains to configure the signal processing algorithm for an autonomous vehicle collision avoidance application. Suggested improvements and ideas for future research efforts for 2D processing, modeling, and an AFIT RNR collision avoidance application are presented here.

5.4.1 2D Processing Future Work.

This research effort determined that real-time 2D processing is not currently feasible, but as technology evolves, the goal of real-time simultaneous time domain range-velocity processing may be within reach. Areas for future study focused on reaching this goal include:

- Determine the feasibility of using a binary ADC in a multiple target environment through simulations and experimental tests.

- Investigate template playback strategies, thus removing reference signal generation and shortening overall 2D processing time.
- Compile the 2D processing algorithm and port the compiled code onto an FPGA to miniaturize the hardware footprint.

5.4.2 RNR Model Future Work.

Some suggestions for updating and improving the AFIT RNR Simulink® model include:

- Update the model to include targets of varying sizes and shapes. Compare the model results to measured results.
- Test model accuracy by comparing measured results in a multiple target environment to model results in the simulated multiple target environment.
- Take the same measurements presented in Section 3.5.2 in an alternate environment (*e.g.* outdoors) to confirm model robustness and accuracy.
- Include a model subsystem to account for clutter and other external environmental factors that affect the radar return signal.

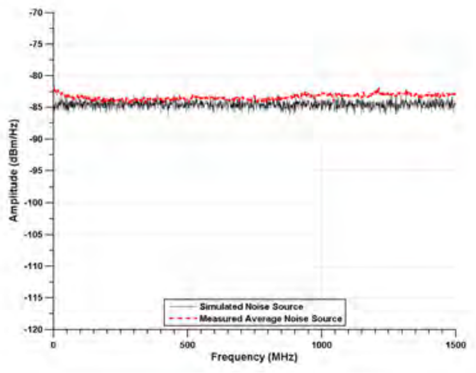
5.4.3 Collision Avoidance Future Work.

There are prime areas for future research that fall outside of the primary research objectives, but are required to use the AFIT RNR as a collision avoidance sensor on an autonomous vehicle:

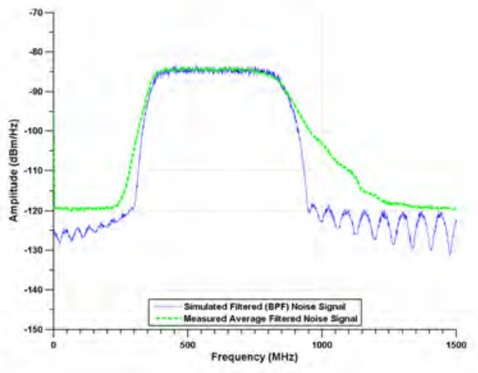
- Develop a collision avoidance algorithm to include search, awareness, and avoidance modes [14]. Determine decision and maneuver time based on RNR detection range, vehicle speed, and obstacle speed, size, and location.

- Select an autonomous vehicle type and develop the interface between sensor algorithm and vehicle control software.
- Determine the impact of clutter on obstacle identification.

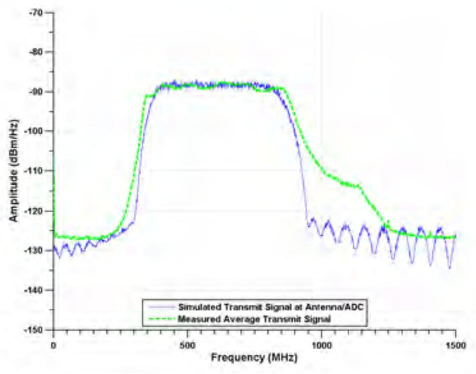
Appendix A. Simulink Model Results



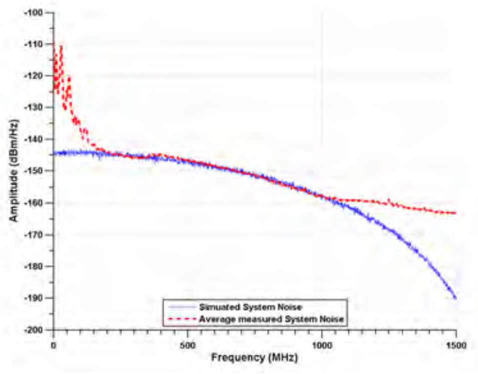
(a) Noise source, TP1



(b) Filtered noise source, TP2

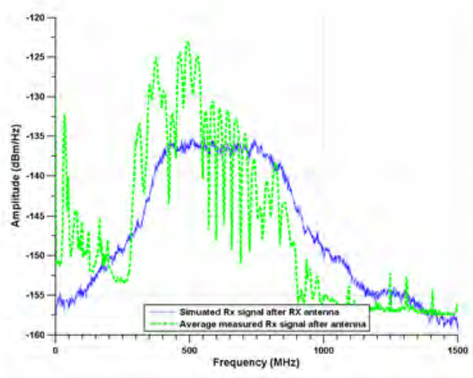


(c) Noise signal at TX antenna, TP3

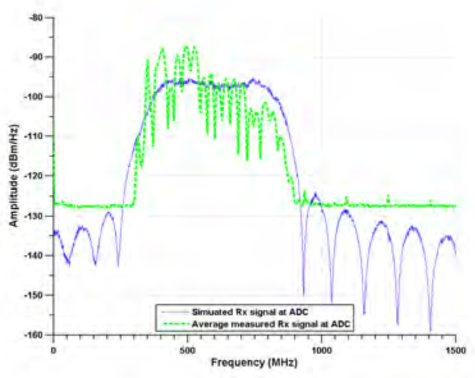


(d) System noise measured with RF off, TP5

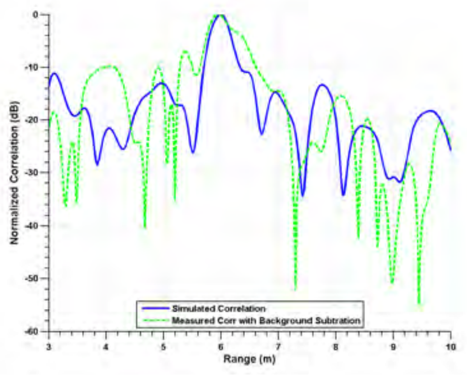
Figure A.1. The measured signals, averaged across 100 realizations, of RNR unit 3 are compared to the simulated signals.



(a) Receive signal after RX antenna, TP4

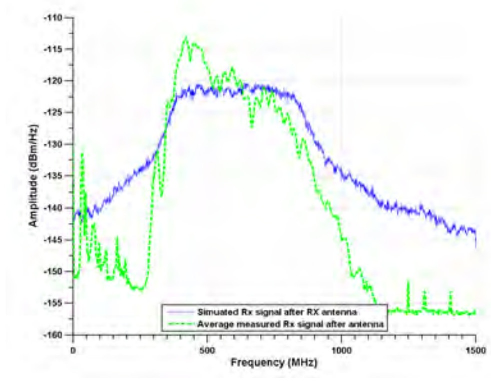


(b) Receive signal at ADC, TP5

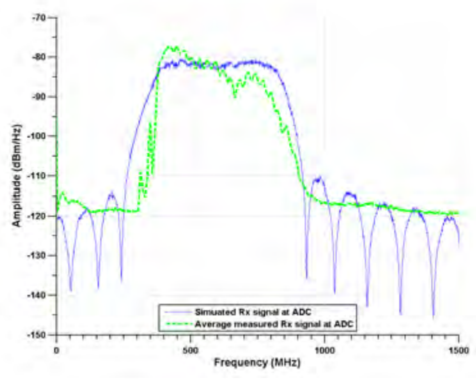


(c) Correlation results

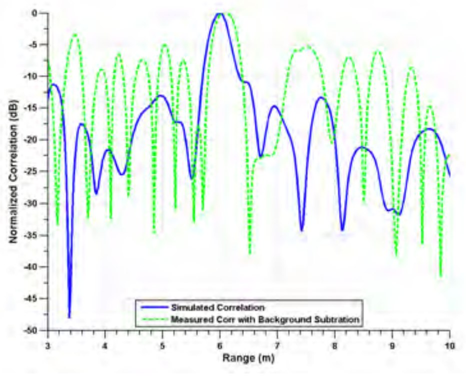
Figure A.2. The (a) and (b) measured receive path signals, averaged across 100 realizations, are compared to the simulated signals. The (c) measured correlation result is compared to the simulated correlation result. All three plots depict results from RNR unit 3, with a 6 meter target and HH polarization.



(a) Receive signal after RX antenna, TP4

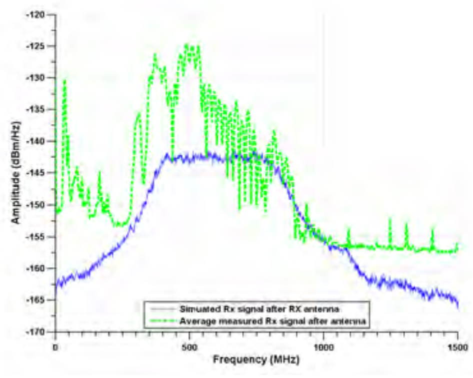


(b) Receive signal at ADC, TP5

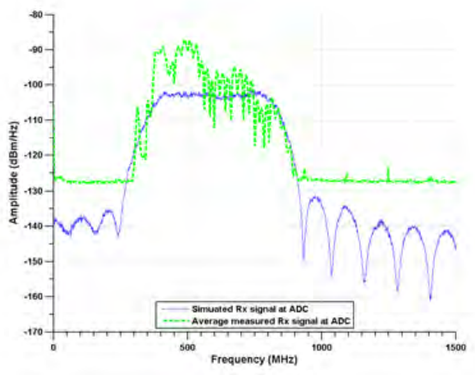


(c) Correlation results

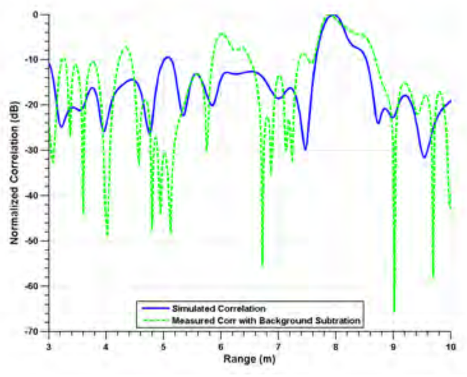
Figure A.3. The (a) and (b) measured receive path signals, averaged across 100 realizations, are compared to the simulated signals. The (c) measured correlation result is compared to the simulated correlation result. All three plots depict results from RNR unit 3, with a 6 meter target and VV polarization.



(a) Receive signal after RX antenna, TP4

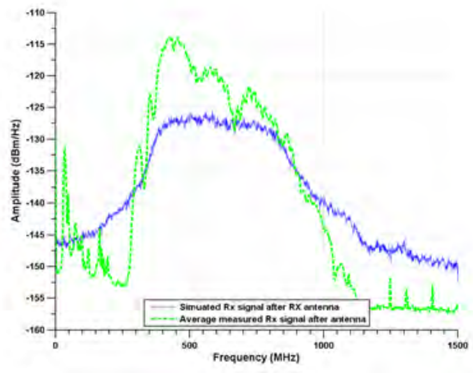


(b) Receive signal at ADC, TP5

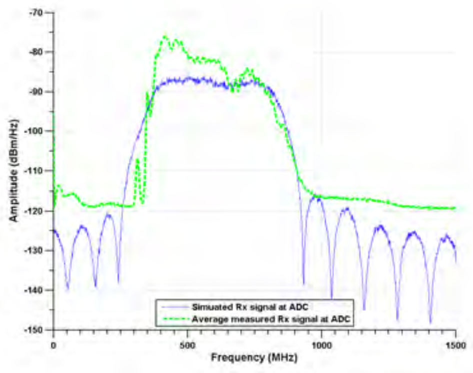


(c) Correlation results

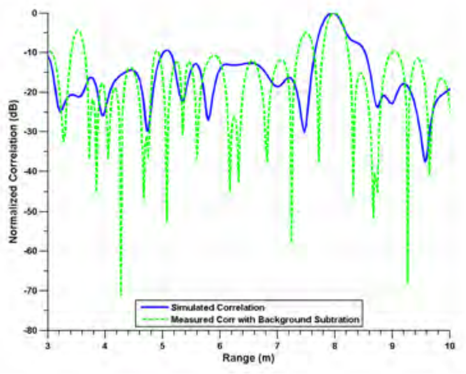
Figure A.4. The (a) and (b) measured receive path signals, averaged across 100 realizations, are compared to the simulated signals. The (c) measured correlation result is compared to the simulated correlation result. All three plots depict results from RNR unit 3, with a 8 meter target and HH polarization.



(a) Receive signal after RX antenna, TP4

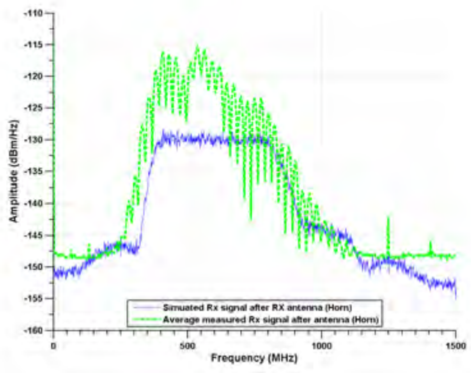


(b) Receive signal at ADC, TP5

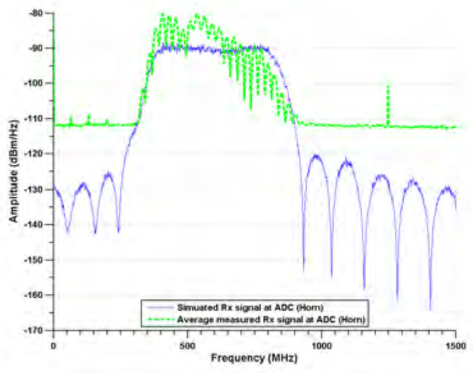


(c) Correlation results

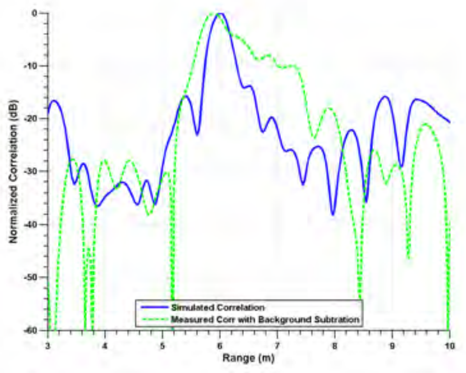
Figure A.5. The (a) and (b) measured receive path signals, averaged across 100 realizations, are compared to the simulated signals. The (c) measured correlation result is compared to the simulated correlation result. All three plots depict results from RNR unit 3, with a 8 meter target and VV polarization.



(a) Receive signal after RX antenna, TP4

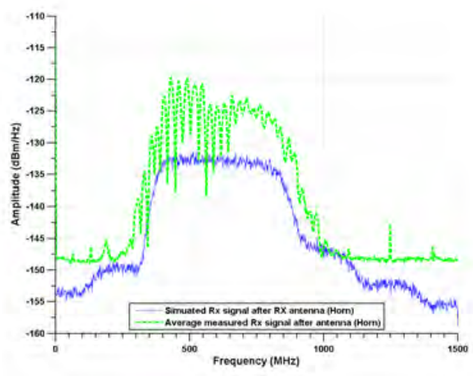


(b) Receive signal at ADC, TP5

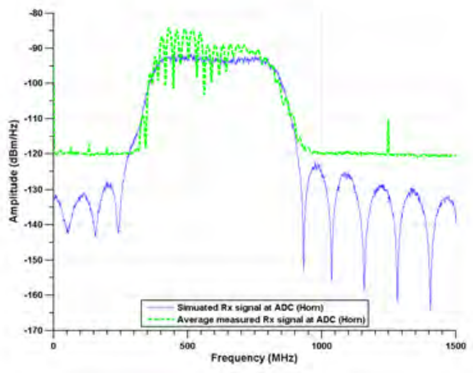


(c) Correlation results

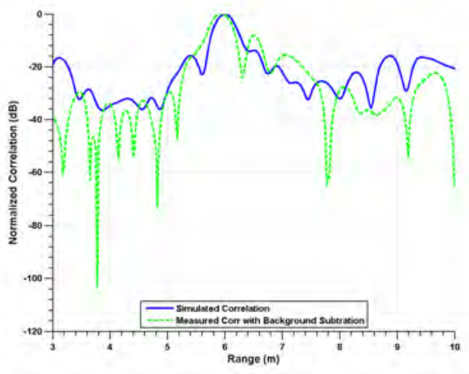
Figure A.6. The (a) and (b) measured receive path signals, averaged across 100 realizations, are compared to the simulated signals. The (c) measured correlation result is compared to the simulated correlation result. All three plots depict results from RNR unit 3 using standard gain horns in place of the LPAs, with a 6 meter target and HH polarization.



(a) Receive signal after RX antenna, TP4

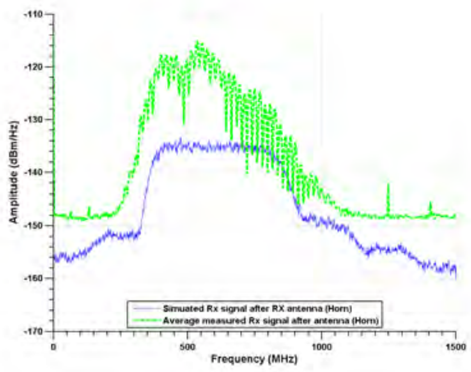


(b) Receive signal at ADC, TP5

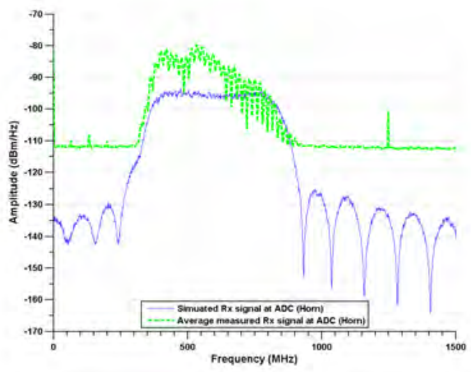


(c) Correlation results

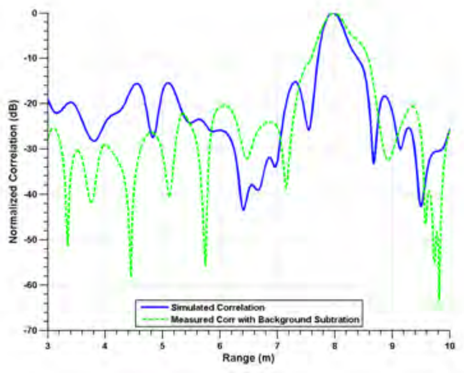
Figure A.7. The (a) and (b) measured receive path signals, averaged across 100 realizations, are compared to the simulated signals. The (c) measured correlation result is compared to the simulated correlation result. All three plots depict results from RNR unit 3 using standard gain horns in place of the LPAs, with a 6 meter target and VV polarization.



(a) Receive signal after RX antenna, TP4

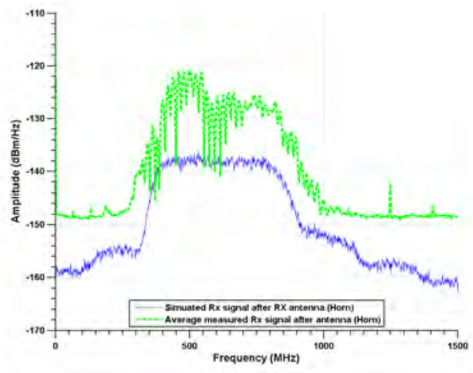


(b) Receive signal at ADC, TP5

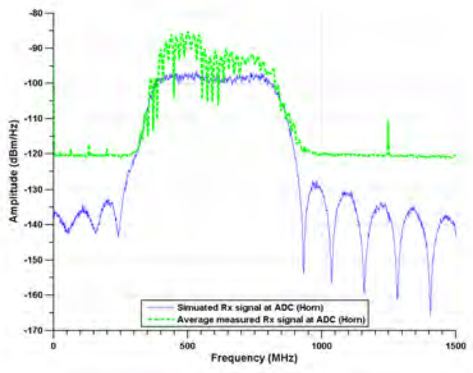


(c) Correlation results

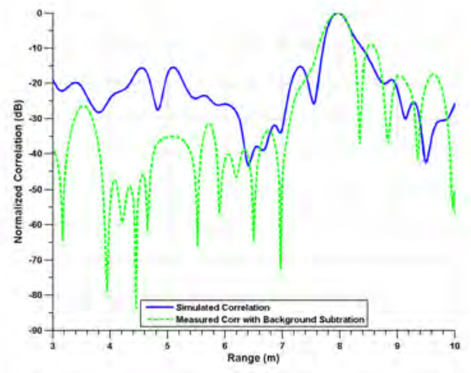
Figure A.8. The (a) and (b) measured receive path signals, averaged across 100 realizations, are compared to the simulated signals. The (c) measured correlation result is compared to the simulated correlation result. All three plots depict results from RNR unit 3 using standard gain horns in place of the LPAs, with a 8 meter target and HH polarization.



(a) Receive signal after RX antenna, TP4

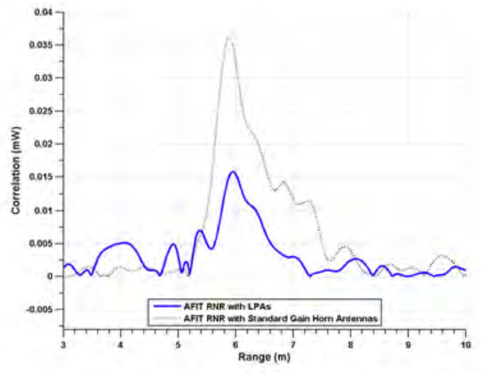


(b) Receive signal at ADC, TP5

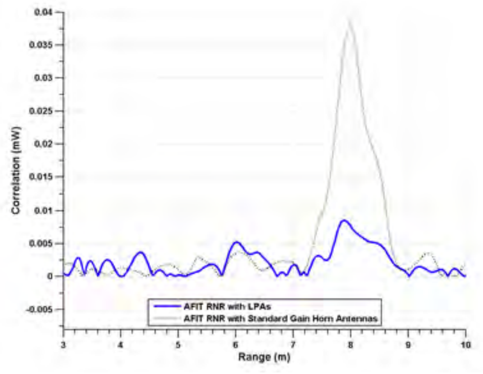


(c) Correlation results

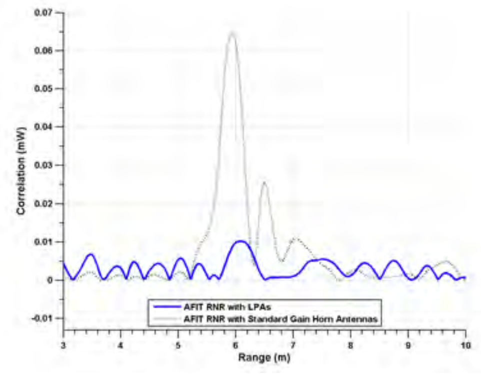
Figure A.9. The (a) and (b) measured receive path signals, averaged across 100 realizations, are compared to the simulated signals. The (c) measured correlation result is compared to the simulated correlation result. All three plots depict results from RNR unit 3 using standard gain horns in place of the LPAs, with a 8 meter target and VV polarization.



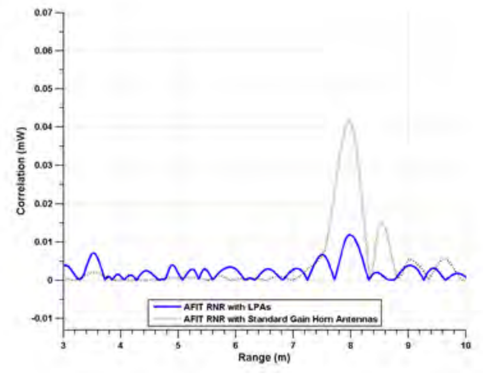
(a) Target at 6 m, HH Polarization



(b) Target at 8 m, HH Polarization

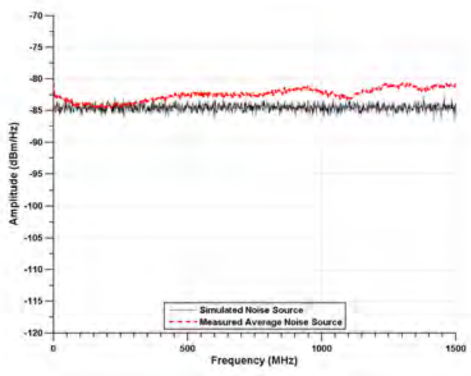


(c) Target at 6 m, VV Polarization

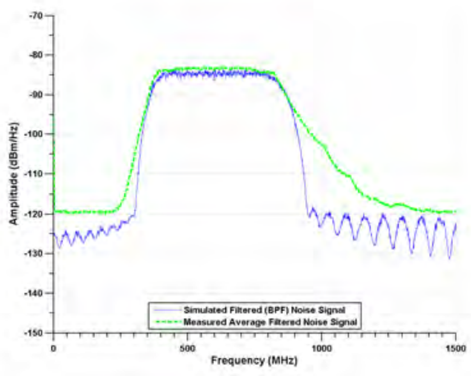


(d) Target at 8 m, VV Polarization

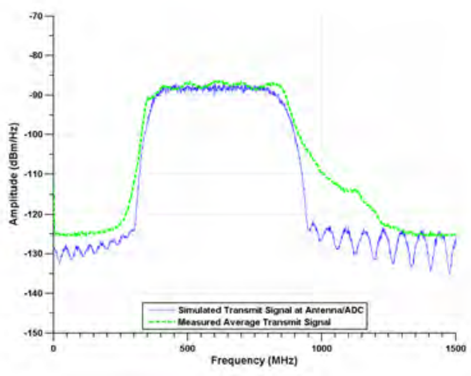
Figure A.10. The receiver output (correlation results) of the AFIT RNR in its standard configuration with LPAs compared to the receiver output of the AFIT RNR using standard gain horns in place of the LPAs.



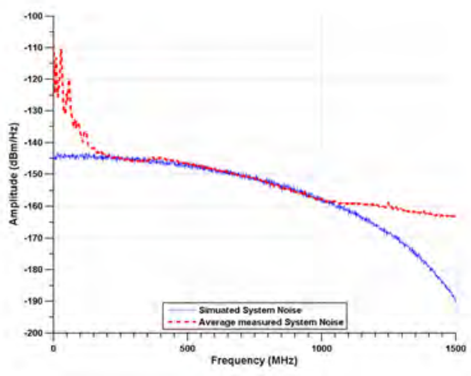
(a) Noise source, TP1



(b) Filtered noise source, TP2

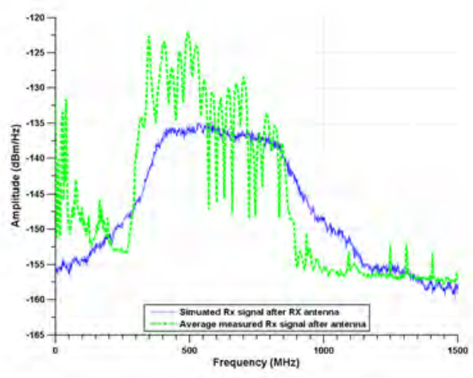


(c) Noise signal at TX antenna, TP3

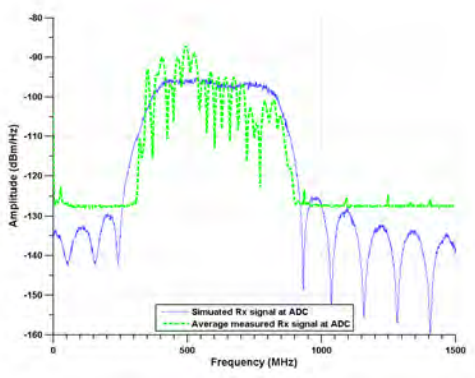


(d) System noise measured with RF off, TP5

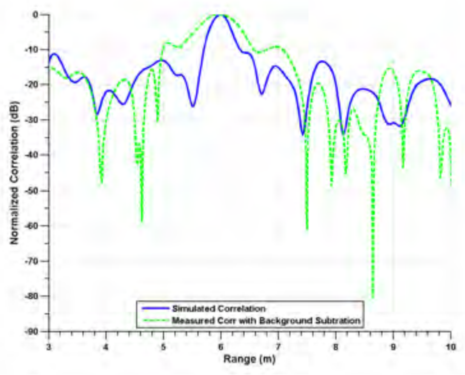
Figure A.11. The measured signals, averaged across 100 realizations, of RNR unit 5 are compared to the simulated signals.



(a) Receive signal after RX antenna, TP4

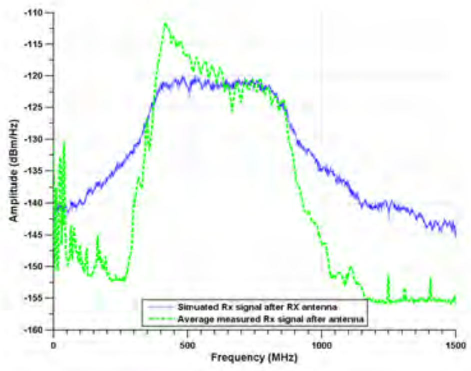


(b) Receive signal at ADC, TP5

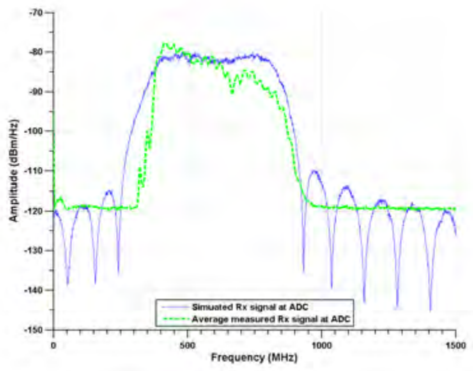


(c) Correlation results

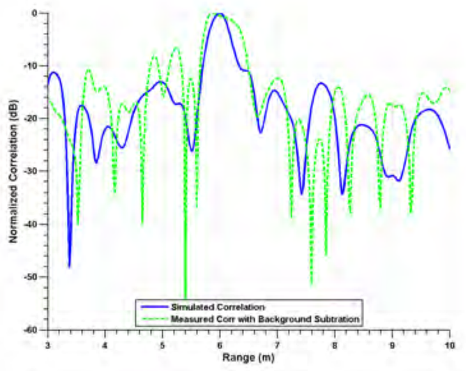
Figure A.12. The (a) and (b) measured receive path signals, averaged across 100 realizations, are compared to the simulated signals. The (c) measured correlation result is compared to the simulated correlation result. All three plots depict results from RNR unit 5, with a 6 meter target and HH polarization.



(a) Receive signal after RX antenna, TP4

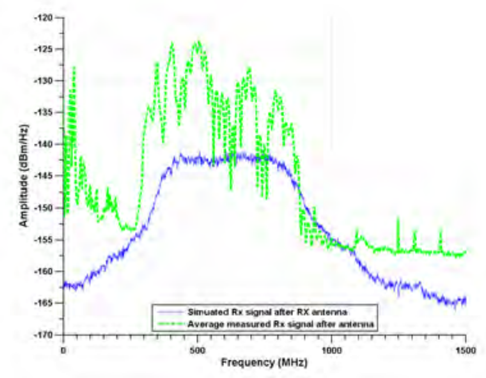


(b) Receive signal at ADC, TP5

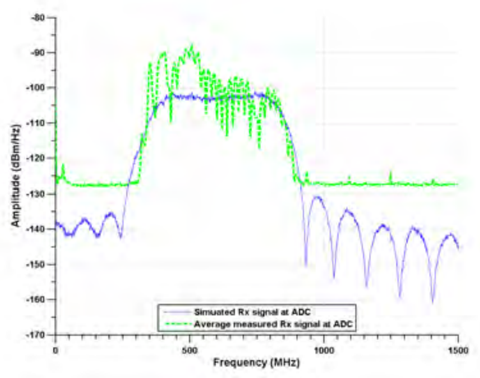


(c) Correlation results

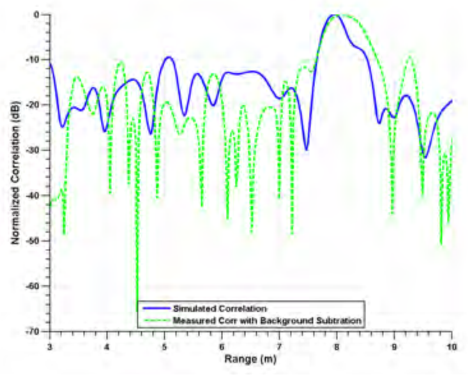
Figure A.13. The (a) and (b) measured receive path signals, averaged across 100 realizations, are compared to the simulated signals. The (c) measured correlation result is compared to the simulated correlation result. All three plots depict results from RNR unit 5, with a 6 meter target and VV polarization.



(a) Receive signal after RX antenna, TP4

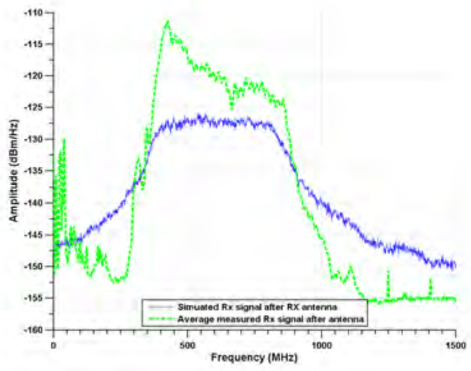


(b) Receive signal at ADC, TP5

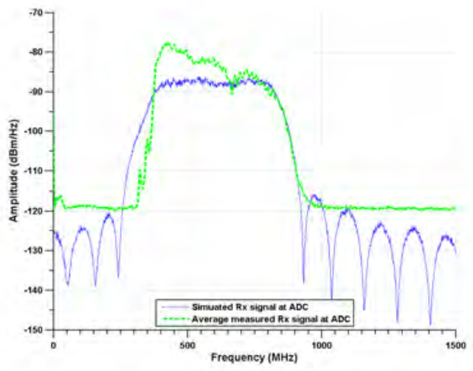


(c) Correlation results

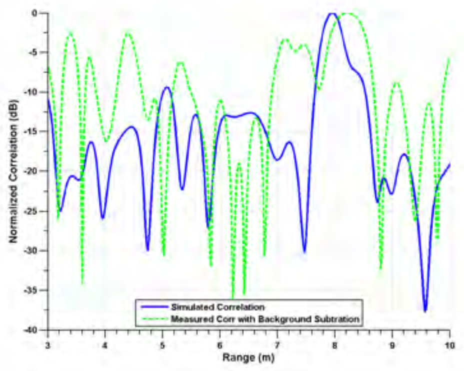
Figure A.14. The (a) and (b) measured receive path signals, averaged across 100 realizations, are compared to the simulated signals. The (c) measured correlation result is compared to the simulated correlation result. All three plots depict results from RNR unit 5, with a 8 meter target and HH polarization.



(a) Receive signal after RX antenna, TP4



(b) Receive signal at ADC, TP5



(c) Correlation results

Figure A.15. The (a) and (b) measured receive path signals, averaged across 100 realizations, are compared to the simulated signals. The (c) measured correlation result is compared to the simulated correlation result. All three plots depict results from RNR unit 5, with a 8 meter target and VV polarization.

Appendix B. MATLAB Code

Listing B.1. This function performs 2D range and velocity processing of the AFIT RNR.

```
function [corr, range] = ...
    process2D(tx, rx, vref, Sample_Frequency, Rmax, GPU)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% This function performs 2 dimensionsal (range and velocity)
% processing of the AFIT Random Noise Radar (RNR) signal,
% simulating the normalized matched filter ouput.  Used to
% post-process Lievsay's data measurement at 1.25 GSamp/sec with
% an inbound target at -5 m/s at at 10 m range.
%
% Authored by Capt Joel Thorson (modified from Capt Lievsay)
% 24 October 2011
%
%  Inputs:
%   tx - Raw data from transmit channel
%   rx - Raw data from receive channel
%   vref - vector of reference velocities for correlation
%   Sample_Frequency - Should be either 1.25e9 or 2.5 e9 GS/s
%   Rmax - Maximum range in meters for correlation
%   GPU - if 0, no GPU; if 1, MATLAB GPU; if 2, Jacket. Default 0
%
%  Output:
%   corr - results of correlation across all reference velocities
%   range - range vector corresponding to range axis of corr
%
%  Other functions required:
%   interpFast() - generates reference signals/no parallelization
%     -and-
%    correlate_jacket() - performs cross corr on GPU using Jacket
%
%     -or-
%    interpFaster() - gnerates ref signals in matrix form/no GPU
%     -and-
%    correlate_no_loop() - faster version/no GPU
%
%     -or-
%    interpFaster_GPU() - generates ref signals in mx form on GPU
%     -and-
%    correlate_no_loop_GPU() - faster, GPU using MATLAB interface
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% Set Parameters
c = 299792458;              %Speed of light
fs = Sample_Frequency;
Number_of_Reference_Velocities = length(vref);

N = length(tx);             %Length of signal
d = c/fs;                   %Distance = Rate * Time
L = round(2*Rmax/d);        %Finds the number of range bins
range=(1:L)*d/2;            %Range vec; saved/used for plotting
G = round(N/L);             %Number of segments to split fft

if GPU ≠ 2                 %Only jacket uses rx in vector form
    rx = reshape(rx,L,G);   %Reshape rx into mx of vecs, length L
    rx = [rx; zeros(L,G,'single')];    %Can't zero pad on GPU fft
end
%% Cycle through all Reference Velocities
tic
```

```
corr=zeros(Number_of_Reference_Velocities,L);    %Preallocate
h = waitbar(0,'Cycling through each reference velocity...');
for ii=1:Number_of_Reference_Velocities

    if GPU == 1                           %use MATLAB's GPU interface
        % build reference signal
        g_sigref = interpFaster_GPU(tx,vref(ii),fs,L);
        % calculate cross correlation function from segments
        corr(ii,:) = correlate_no_loop_GPU(g_sigref,rx,L,G);
        clear g_sigref

    elseif GPU == 2                       %use Jacket's GPU interface
        % build reference signal
        sigref = interpFast(tx,vref(ii),fs);
%         g_sigref = interpFaster_jacket(tx,vref(ii),fs,L);
        % calculate cross correlation function from segments
        corr(ii,:) = correlate_jacket(sigref,rx,L,G);
%         corr(ii,:) = correlate_jacket_faster(g_sigref,rx,L,G);

    else                                  %No GPU interface
        % build reference signal
        sigref = interpFaster(tx,vref(ii),fs,L);
        % calculate cross correlation function from segments
        corr(ii,:) = correlate_no_loop(sigref,rx,L,G);
    end

    % Display waitbar and # reference velocities completed
    fprintf('%d of %d Completed\n',ii,...
        Number_of_Reference_Velocities);
    waitbar(ii/Number_of_Reference_Velocities,h);
    toc
end
```

```
close(h)

return
```

Listing B.2. This function generates a reference signal bank using vectors without parallelization.

```
function [sigref] = interpFast(tx,vref,fs)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% This function replaces the interp1 function for linear
% interpolation of AFIT's random noise radar signal, used to
% build a reference signal bank.  It is simply an index shift to
% dialate the transmit signal according to a specified velocity,
% while maintaing sample rate.
%
% Inputs:
%   tx - signal that was tranmitted
%   vref - velocity that determines dialation of transmit signal
%   fs - sampling frequency
%
% Output:
%   sigref - The reference signal used for correlation
%
% Author: Capt T. Joel Thorson - 22 September 2011
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

c=299792458;                                  %Speed of light
Ts = 1/fs;                                    %Time between samples

L = length(tx);
delt=2*vref/((c-vref)*fs);      %Per sample time shift (const v.)
N = ceil(L*abs(delt/Ts));       %max time shift for given velocity
txshift = zeros(N+1,1);         %allocate vector of shift amounts
remainder = zeros(1,N);         %allocate overflow tracker
sigref = ones(L,1,'single');    %set to all ones to start
```

```
if vref < 0    %For Negative Reference Velocities
    for n = 1:N+1
        remainder(n) = (n-1)*((Ts/abs(delt) - Ts) - ...
            floor(Ts/abs(delt)-Ts));
            %Tracks overflow to determine when to add shift
        txshift(n+1) = ceil(Ts/abs(delt)/2) + ...
            (n-1)*(floor(Ts/abs(delt)-Ts)) + floor(remainder(n));
            %Determine when (what sample) to shift signal
        if txshift(n+1)+n-1 > L
            sigref(txshift(n)+1:L-n+1) = ...   %sigref is shifted
                tx(txshift(n)+n:L);            %version of tx
        else
            sigref(txshift(n)+1:txshift(n+1)) =...%sigref shifted
                tx(txshift(n)+n:txshift(n+1)+n-1);%version of tx
        end
    end

elseif vref==0  %No change to signal
    sigref=tx;

else    %For Positive Reference Velocities
    for n = 1:N+1
    remainder(n) = (n-1)*((Ts/abs(delt) + Ts) - ...
        floor(Ts/abs(delt)+Ts));
    txshift(n+1) = ceil(Ts/abs(delt)/2 + Ts) + ...
        (n-1)*(floor(Ts/abs(delt) + Ts)) + floor(remainder(n));
        %Determine when (what sample) to shift signal
        if txshift(n+1) > L
            sigref(txshift(n)+1:L) = ...       %sigref is shifted
               tx(txshift(n)-n+2:L-n+1);       %version of tx
        else
            sigref(txshift(n)+1:txshift(n+1)) =...%sigref shifted
```

```
               tx(txshift(n)-n+2:txshift(n+1)-n+1);%version of tx
        end
    end
end
return
```

Listing B.3. This function generates a reference signal bank in matrix form without parallelization. It is faster than interpFast but requires more memory.

```
function [sigref] = interpFaster(tx,vref,fs,L)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% This function replaces the interp1 function for linear
% interpolation of AFIT's random noise radar signal, used to
% build a reference signal bank.  It is simply an index shift to
% dialate the transmit signal according to a specified velocity,
% while maintaing sample rate.
%
% Inputs:
%   tx - signal that was tranmitted
%   vref - velocity that determines dialation of transmit signal
%   fs - sampling frequency
%   L - number of range bins
%
% Output:
%   sigref - The reference signal used for correlation in the
%   matrix format required for use in the correlate_no_loop
%   function.  This is a LxG matrix on the CPU in the format
%   required for correlate_no_loop()
%
% Author: Capt T. Joel Thorson - 24 October 2011
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

c=299792458;                  %Speed of light
Ts = 1/fs;                    %Time between samples

N = length(tx);
G = N/L;                      %Number of segments
delt=2*vref/((c-vref)*fs);    %Per sample time shift
```

```
K = ceil(N*abs(delt/Ts));      %max time shift for given velocity
% tic;
sigref = zeros(L,N/L,'single');         %set to all ones to start
% toc;
if vref < 0                    %For Negative Reference Velocities
    remainder = ((1:K+1)-1).*((Ts/abs(delt) - Ts) - ...
        floor(Ts/abs(delt)-Ts));
    shift_index = ceil(Ts/abs(delt)/2)+((1:K+1)-1).*...
        (floor(Ts/abs(delt)-Ts)) + floor(remainder);
    shift_index(shift_index>N) = []; %don't shift > signal length
    K = length(shift_index);
    extra = mod(shift_index(1),L);
    column = floor(shift_index(1)/L);
    segment = tx(1:(shift_index(1)-extra));
    sigref(:,1:column) = reshape(segment,L,column);

    for k = 2:K
        beginning = L-extra;
        compress = k-1;
        sigref(:,column+1) = ...
            [tx(shift_index(k-1)-extra+1:shift_index(k-1));...
            tx(shift_index(k-1)+compress+1:shift_index(k-1)+...
            compress+beginning)];
        column_prev = column+1;
        extra = mod(shift_index(k)-shift_index(k-1)-beginning,L);
        column = floor(shift_index(k)/L);
        segment = tx(shift_index(k-1)+compress+beginning+...
            1:shift_index(k)+compress-extra);
        sigref(:,column_prev+1:column) = ...
            reshape(segment,L,column-column_prev);
    end
    %From max tx_shift to end...
    beginning = L-extra;
    compress = K;
    sigref(:,column+1) = ...
        [tx(shift_index(K)-extra+1:shift_index(K));...
        tx(shift_index(K)+compress+1:shift_index(K)+...
        compress+beginning)];
    column_prev = column+1;
    segment = tx(shift_index(K)+compress+beginning+1:end);
    columns = floor(length(segment)/L);
    sigref(:,column_prev+1:column_prev+columns) =...
        reshape(segment(1:columns*L),L,columns);

elseif vref==0  %No change to signal
    sigref=reshape(tx,L,G);

else    %For Positive Reference Velocities
    remainder = ((1:K+1)-1).*((Ts/abs(delt) + Ts) - ...
        floor(Ts/abs(delt)+Ts));
    shift_index = ceil(Ts/abs(delt)/2+Ts)+((1:K+1)-1).*...
        (floor(Ts/abs(delt)+Ts)) + floor(remainder);
    shift_index(shift_index>N) = []; %don't shift > signal length
    K = length(shift_index);
    extra = mod(shift_index(1),L);
    column = floor(shift_index(1)/L);
    segment = tx(1:(shift_index(1)-extra));
    sigref(:,1:column) = reshape(segment,L,column);

    if K > 1;
        for k = 2:K
            beginning = L-extra;
            stretch = k-1;
            sigref(:,column+1) = ...
               [tx(shift_index(k-1)-extra+1:shift_index(k-1));...
               tx(shift_index(k-1)-stretch+1:shift_index(k-1)-...
               stretch+beginning)];
            column_prev = column+1;
            extra = ...
                mod(shift_index(k)-shift_index(k-1)-beginning,L);
            column = floor(shift_index(k)/L);
            segment = tx(shift_index(k-1)-stretch+beginning+...
                1:shift_index(k)-stretch-extra);
            sigref(:,column_prev+1:column) = ...
                reshape(segment,L,column-column_prev);
        end
    end
    %From max tx_shift to end...
    beginning = L-extra;
    stretch = K;
    sigref(:,column+1) = ...
        [tx(shift_index(K)-extra+1:shift_index(K));...
        tx(shift_index(K)-stretch+1:shift_index(K)-...
        stretch+beginning)];
    column_prev = column+1;
    columns = G-column_prev; %Columns left that need to be filled
    segment = tx(shift_index(K)-stretch+beginning+1:...
        shift_index(K)-stretch+beginning+columns*L);
    sigref(:,column_prev+1:G) = reshape(segment,L,columns);

end
return
```

Listing B.4. This function generates a reference signal bank using the GPU.

```
function [g_sigref] = interpFaster_GPU(tx,vref,fs,L)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% This function replaces the interp1 function for linear
% interpolation of AFIT's random noise radar signal, used to
% build a reference signal bank.  It is simply an index shift to
% dialate the transmit signal according to a specified velocity,
% while maintaing sample rate.  Uses Matlab's GPU interface.
%
% Inputs:
%   tx - signal that was tranmitted
%   vref - velocity that determines dialation of transmit signal
%   fs - sampling frequency
%   L - number of range bins
%
% Output:
%   g_sigref - The reference signal used for correlation in the
%   matrix format required for use in the correlate_no_loop
%   function.  This is a LxG matrix on the GPU in the format
%   required for correlate_no_loop_faster()
%
% Author: Capt T. Joel Thorson - 24 October 2011
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

c=299792458;                        %Speed of light
Ts = 1/fs;                          %Time between samples

N = length(tx);
G = N/L;                            %Number of segments
delt=2*vref/((c-vref)*fs);          %Per sample time shift
K = ceil(N*abs(delt/Ts));           %max time shift for given velocity
```

```
if vref < 0                        %For Negative Reference Velocities
    g_sigref = parallel.gpu.GPUArray.zeros...
        (L,N/L,'single');          %allocate on GPU
    remainder = ((1:M)-1).*((Ts/abs(delt) - Ts) - ...
        floor(Ts/abs(delt)-Ts));
    shift_index = ceil(Ts/abs(delt)/2)+((1:M)-1).*...
        (floor(Ts/abs(delt)-Ts)) + floor(remainder);
    shift_index(shift_index>N) = [];           %don't shift > length
    K = length(shift_index);
    extra = mod(shift_index(1),L);
    column = floor(shift_index(1)/L);
    g_segment = gpuArray(tx(1:(shift_index(1)-extra)));
    g_sigref(:,1:column) = reshape(g_segment,L,column);
    clear g_segment

    for k = 2:K
        beginning = L-extra;
        compress = k-1;
        g_sigref(:,column+1) = [gpuArray(tx(shift_index(k-1)-...
            extra+1:shift_index(k-1)));...
            gpuArray(tx(shift_index(k-1)+compress+...
            1:shift_index(k-1)+compress+beginning))];
        column_prev = column+1;
        extra = mod(shift_index(k)-shift_index(k-1)-beginning,L);
        column = floor(shift_index(k)/L);
        g_segment = gpuArray(tx(shift_index(k-1)+compress+...
            beginning+1:shift_index(k)+compress-extra));
        g_sigref(:,column_prev+1:column) = ...
            reshape(g_segment,L,column-column_prev);
        clear g_segment
    end
    %From max tx_shift to end...
    beginning = L-extra;
    compress = K;
    g_sigref(:,column+1) = [gpuArray(tx(shift_index(K)-extra+...
        1:shift_index(K)));...
        gpuArray(tx(shift_index(K)+compress+1:shift_index(K)+...
        compress+beginning))];
    column_prev = column+1;
    g_segment  = ...
        gpuArray(tx(shift_index(K)+compress+beginning+1:end));
    columns = gather(floor(length(g_segment)/L));
    g_sigref(:,column_prev+1:column_prev+columns) = ...
        reshape(g_segment(1:columns*L),L,columns);
    clear g_segment

elseif vref==0  %No change to signal
    g_sigref = reshape(gpuArray(tx),L,G);


else    %For Positive Reference Velocities
    g_sigref = parallel.gpu.GPUArray.zeros...
        (L,N/L,'single');                    %allocate on GPU
    remainder = ((1:M)-1).*((Ts/abs(delt) + Ts) - ...
        floor(Ts/abs(delt)+Ts));
    shift_index = ceil(Ts/abs(delt)/2+Ts)+((1:M)-1).*...
        (floor(Ts/abs(delt)+Ts)) + floor(remainder);
    shift_index(shift_index>N) = [];        %don't shift > length
    K = length(shift_index);
    extra = mod(shift_index(1),L);
    column = floor(shift_index(1)/L);
    g_segment = gpuArray(tx(1:(shift_index(1)-extra)));
    g_sigref(:,1:column) = reshape(g_segment,L,column);
```

```
clear g_segment

if K > 1;
    for k = 2:K
        beginning = L-extra;
        stretch = k-1;
        g_sigref(:,column+1) = ...
            [gpuArray(tx(shift_index(k-1)-extra+...
            1:shift_index(k-1)));gpuArray(tx...
            (shift_index(k-1)-stretch+1:shift_index(k-1)-...
            stretch+beginning))];
        column_prev = column+1;
        extra = ...
            mod(shift_index(k)-shift_index(k-1)-beginning,L);
        column = floor(shift_index(k)/L);
        g_segment = gpuArray(tx(shift_index(k-1)-stretch+...
            beginning+1:shift_index(k)-stretch-extra));
        g_sigref(:,column_prev+1:column) = ...
            reshape(g_segment,L,column-column_prev);
        clear g_segment
    end
end
%From max tx_shift to end...
beginning = L-extra;
stretch = K;
g_sigref(:,column+1) = [gpuArray(tx(shift_index(K)-extra+...
    1:shift_index(K)));...
    gpuArray(tx(shift_index(K)-stretch+1:shift_index(K)-...
    stretch+beginning))];
column_prev = column+1;
columns = G-column_prev; %Columns left that need to be filled
g_segment = gpuArray(tx(shift_index(K)-stretch+beginning+...
            1:shift_index(K)-stretch+beginning+columns*L));
        g_sigref(:,column_prev+1:G) = reshape(g_segment,L,columns);
        clear g_segment

    end
    return
```

Listing B.5. This function performs cross correlation without using a GPU.

```
function [corr] = correlate_no_loop(sigref,rx,L,G)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% This function performs the cross correlation of the received
% signal with the generated reference signal for a given
% velocity.  It splits the signal into bite-size chunks in order
% to parallelize the fft processing. The construct for splitting
% the fft came from an article by Dr. Michal Meller titled
% "Some Aspects of Designing Real-Time Digital Correlators
% for Noise Radars." 2010 IEEE
%
% Inputs:
%   sigref - reference signal generated using interpFast()
%   rx - receive signal
%   L - number of range bins
%   G - number of segments to split signals for fft processing
%
% Output:
%   corr - correlation results with L range bins for given
%          reference signal
%
% Author: Capt T. Joel Thorson - 18 October 2011
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% Reshape rx and sigref into matices
% The matrices will consist of G vectors of lenght 2*L.  The
% first L in sigref will consist of the previous segment and the
% second L will consist of the "current" segment.  The first L of
% the rx vectors consist of the current segment and the second L
% will consist of zeros.
```

```
sigref_mx_bottom = reshape(sigref,L,G); %bottom half of mx
sigref_mx_top = [zeros(L,1) sigref_mx_bottom(:,1:G-1)]; %top half
sigref_mx = [sigref_mx_top; sigref_mx_bottom];  %reshaped matrix
clear sigref_mx_*

%% Perform correlation on matrices using FFT.
% Ideally all G vectors will be correlated at the same time,
% requiring a simple sum at the end to produce the results, but
% there is not enough memory in the GPU, so it must split

sigref_fft = fft(sigref_mx);
clear sigref_mx
rx_fft = fft(rx);
clear rx
dual_fft = conj(sigref_fft).*rx_fft;
clear sigref_fft rx_fft
temp = ifft(dual_fft);
clear dual_fft
temp = temp(L+1:end,:);
corr = sum(temp,2).';
return
```

Listing B.6. This function performs cross correlation on the GPU using the Jacket® interface.

```
function [corr] = correlate_jacket(sigref,rx,Rmax,G)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% This function performs the cross correlation of the received
% signal with the generated reference signal for a given
% velocity.  It splits the signal into bite-size chunks in order
% to parallelize the fft processing. The construct for splitting
% the fft came from an article by Dr. Michal Meller titled
% "Some Aspects of Designing Real-Time Digital Correlators
% for Noise Radars." 2010 IEEE
%
% Inputs:
%   sigref - reference signal generated using interpFast()
%   rx - receive signal
%   Rmax - number of samples equivalent to maximum range
%   G - number of segments to split signals for fft processing
%
% Output:
%   sigref - The reference signal used for correlation
%
% Author: Capt T. Joel Thorson - 28 September 2011
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clear g_* gpu_hook

nfft = 2*Rmax;                              %number of points in the fft
% Calculate first seg on CPU (next segs require prior seg info)
rxshort = rx(1:Rmax);                       %only 1 segment worth
sigrefshort = [zeros(Rmax,1);sigref(1:Rmax)];
temp = ifft(conj(fft(sigrefshort)).*...
    fft(rxshort,nfft)).';                   %Correlation
```

```
% Create GPU Variables
g_Rmax = gsingle(Rmax);

% create a vec of indices for use in the gfor loop--Jacket req't
g_rx_index_vec = gsingle(1:Rmax);
g_ref_index_vec = gsingle(1:2*Rmax);

%% Execute a Double for loop
% The outer loop breaks the sigref and rx signals into smaller
% sections to avoid overloading the memory on the GPU.
% The inner loop is the GPU for loop.  It is designed to quickly
% process the correlation in small FFT segments of length 2*Rmax
% in parallel across hundreds of processing cores on the GPU.

%Create outer loop size - needs to be a multiple of segment size
stepsize = 400*2*Rmax;  %Bigger step size = more GPU memory req'd

g_corr_temp = gzeros(G-1,Rmax);
for gg = 1:stepsize:G  %Get sigref/rx signals one chunk at a time
    if gg ≤ G-1-stepsize  %Determine if in the last outside loop
        g_rx = gsingle(rx(gg*Rmax+1 + ...
            (0:stepsize*Rmax-1))); %Segment of rx as GPU variable
        g_sigref = gsingle(sigref((gg-1)*Rmax+1 + ...
            (0:stepsize*Rmax-1)));  %Segment of sigref (that has
            %extra Rmax segment) as GPU variable

        gfor g = gg:gg+stepsize-1  %parallelize FFT in segments
            g_rx_start = (g-gg)*g_Rmax; %Start index for rx seg
            g_ref_start = (g-gg)*g_Rmax; %Start index for ref seg
            g_rxshort = g_rx(g_rx_start+g_rx_index_vec);  %rx seg
            g_sigrefshort = ...    %previous and current segments
```

```
                g_sigref(g_ref_start+g_ref_index_vec);
            g_temp = real(ifft(conj(fft(g_sigrefshort)).*...
                fft(g_rxshort,nfft)).');            %Correlation
            g_corr_temp(g,:) = g_temp(Rmax+1:end);  %half IFFT
        gend
    else  %Have to account for remaining segments
        %Smaller segment of rx as GPU variable
        g_rx = gsingle(rx(end-(G-gg)*Rmax:end));
        g_sigref = gsingle(sigref(end-(G-gg+1)*Rmax:end));

        gfor g = gg:G-1  %parallelize FFT in segments
            g_rx_start = (g-gg)*g_Rmax; %Start index for rx seg
            g_ref_start = (g-gg)*g_Rmax; %Start index for ref seg
            g_rxshort = g_rx(g_rx_start+g_rx_index_vec);  %rx seg
            g_sigrefshort = ...    %previous and current segments
                g_sigref(g_ref_start+g_ref_index_vec);
            g_temp = real(ifft(conj(fft(g_sigrefshort)).*...
                fft(g_rxshort,nfft)).');            %Correlation
            g_corr_temp(g,:) = g_temp(Rmax+1:end);  %half IFFT
        gend
    end
end
corr_temp = [temp(Rmax+1:end);double(g_corr_temp)];%Gather on CPU
corr = sum(corr_temp);              %Accumulate (add the vectors)
clear corr_temp temp g_* gpu_hook
return
```

Listing B.7. This function performs cross correlation on the GPU using MATLAB®'s GPU interface.

```
function [corr] = correlate_no_loop_GPU(g_sigref,rx,L,G)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% This function performs the cross correlation of the received
% signal with the generated reference signal for a given
% velocity.  It splits the signal into bite-size chunks in order
% to parallelize the fft processing. The construct for splitting
% the fft came from an article by Dr. Michal Meller titled
% "Some Aspects of Designing Real-Time Digital Correlators
% for Noise Radars." 2010 IEEE
%
% Inputs:
%   g_sigref - ref signal generated using interpFaster_GPU() (mx)
%   rx - receive signal
%   L - number of range bins
%   G - number of segments signals split for fft processing
%
% Output:
%   corr - correlation results with L range bins for given
%          reference signal
%
% Author: Capt T. Joel Thorson - 18 October 2011
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% tic

%% Reshape rx and sigref into matices
% The matrices will consist of G vectors of lenght 2*L.  The
% first L in sigref will consist of the previous segment and the
% second L will consist of the "current" segment.  The first L of
% the rx vectors consist of the current segment and the second L
% will consist of zeros.

g_sigref_top = [parallel.gpu.GPUArray.zeros(L,1,'single') ...
    g_sigref(:,1:G-1)];                                    %top half
g_sigref_mx = [g_sigref_top; g_sigref];
clear g_sigref g_sigref_top

g_rx = gpuArray(rx);  %Added this here but only works w/ 6 GB GPU

%% Perform correlation on matrices using FFT.
% Ideally all G vectors will be correlated at the same time,
% requiring a simple sum at the end to produce the results, but
% there is not enough memory in the GPU, so it must split

K = 8;          %number of groups to split into
g_corr_temp = parallel.gpu.GPUArray.zeros(L,1,'single');
for k = 1:K
    g_sigref_short = g_sigref_mx(:,(k-1)*G/K+1:k*G/K);
%     g_rx_short = gpuArray(rx(:,(k-1)*G/K+1:k*G/K));
        %Uncomment above when using 4 GB GPU
    g_rx_short = g_rx(:,(k-1)*G/K+1:k*G/K);
        %Comment out above when not using 6 GB GPU
    g_sigref_fft = fft(g_sigref_short);
    clear g_sigref_short
    g_sigref_fft_conj = conj(g_sigref_fft);
    clear g_sigref_fft;
    g_rx_fft = fft(g_rx_short);
    clear g_rx_short
    g_dual_fft = g_sigref_fft_conj.*g_rx_fft;
    clear g_sigref_fft_conj g_rx_fft
    g_temp = ifft(g_dual_fft);
    clear g_dual_fft
    g_temp = g_temp(L+1:end,:);
    g_corr_temp = g_corr_temp + sum(g_temp,2);
    clear g_temp
end
corr = gather(real(g_corr_temp.'));
% toc
return
```

Listing B.8. This function runs the AFIT RNR Simulink® model.

```
function [snr_calc,snr_sim,snr_meas] = ...
    AFIT_RNR(unit,pol,r,time,horn)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Description:  This function simulates the AFIT random noise
%   radar (RNR)by calling a Simulink model.  The Simulink model
%   simulates each of the hardware components in the RNR, the
%   target environment, system noise, and the analog-to-digital
%   conversion.  It then uses the same signal processing
%   architecture as implemented in the AFIT RNR.  The results of
%   the simulations are compared with measured data collected by
%   Capt T. Joel Thorson in November 2011-January, 2012.  The
%   model can simulate more than one target, but only one target
%   was used when collecting measurements. Thus, this function is
%   limited to only a single target.  The model can be modified
%   to simulate moving targets as well, but does not have
%   measurements for comparison.
%
% Inputs:
%   unit - The RNR box number used for measurements (3 or 5)
%   pol - The polarization of antennas used for measurments
%         ('HH' or 'VV')
%   r - The distance from RNR to target used for measurements
%       (6 or 8) m
%   time - Measurement window (1 or 2) for 1.0 and 2.23 microsecs
%   horn - Was the standard gain horn used in place of the LPAs?
%       (0 if LPAs used, 1 if horn used)
%
% Outputs:
%   snr_calc - Calculated (using RRE) SNR at input to ADC
%   snr_sim - Simulated SNR at input to ADC
%   snr_meas - Measured SNR at input to ADC
%
% Subfunctions required:
%   peakfinderN_mod() - used to plot smooth correlation results
%
% Author:
%   Capt T. Joel Thorson, modified from original by
%   Capt John Priestly, 19 January 2012
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% SET MEASUREMENT PARAMETERS (To compare to Model)
%  This can be turned into function inputs later

% unit = 3;       %Took measurements on unit 3 and 5
% pol = 'VV';     %Took measurements using HH and VV polarization
% r = 8;          %Took measurements at r = 6 and 8 meters
% time = 1;       %Either 1 or 2 for 1 or 2.23 microsec collect
% horn = 1;       %horn antenna used? (0 if no, 1 if yes)

if time == 2;
    T = 2.23e-6;    %Set collection/simulation time
else
    time = 1;       %1 microsecond is default and AFIT RNR time
    T = 1e-6;       %set collection/simulation time
end

%% PREPARE SIMULINK MODEL

model = 'AFIT_RNR_model';          %model name
load_system(model)                 %load model
inputs =[];                        %no inputs for simulink model
options = [];                      %no options for simulink model
```

```
warning off all                         %turn off warnings

%% SET INITIAL PARAMETERS

fs = 4e9;                   %Sample freq of analog components in Hz
df = (1/T);                 %Frequency resolution
f = (0:df:fs);              %freq row vector in Hz
f1 = (0:df:fs+3e9)/10^6;    %freq after delay blks change by 3 GHz
L = length(f);

db_offset = pow2db(df); %for conversion from dBm/df to dBm/Hz

rx_fs = 1.5e9;              %sample rate of DCR receiver (ADC)

%% ANTENNA GAIN (Based on Measured S11 by Lt Ludwig)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Note: The signal that propogates through the simulink model is
%   a voltage.  So for all gain blocks, the dB gain must be
%   converted to linear using 10^(G_db/20) or sqrt(db2pow(G_db)).
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


if horn == 0
    % LPA Antenna Gain
    filename = 'LPA_antenna.xls';
    S11 = xlsread(filename);            %Antenna reflection curve
    freq_s11 = S11(:,1);                %Frequencies in Hz
    freq_s11(freq_s11>fs) = [];         %Only want freq < fs
    reflect = S11(1:length(freq_s11),2);%Amplitude in dB
    pass = -reflect; %Assumed (unscaled) transmission through ant
    gain = pass-10;                     %Scaled to avg ¬6 dB
    Gt_db = interp1(freq_s11,gain,f);   %Gain in dB @ each sample
```

```
else
    % Standard Gain Horn Antenna Gain (uncomment as required)
    filename = 'Standard_Gain.xls';
    mx = xlsread(filename);             %freq and gain info
    freq_std = mx(:,1)*1e9;             %Frequencies in Hz
    freq_std(freq_std>fs) = [];         %Only want freq < fs
    gain_std = mx(1:length(freq_std),2);%Gain in dB
    Gt_db = interp1(freq_std,gain_std,f);%Gain in dB @ each samp
    Gt_db = Gt_db;                      %adjust gain
end

% Convert Gain in dB to linear as applied to voltage signal
Gt_lin = sqrt(db2pow(Gt_db));      %Linear gain @ each sampl
Gt_lin(isnan(Gt_lin)) = 0;         %find NaN's and set to 0

% Gt_lin = sqrt(db2pow(6))*ones(1,L);   %uncomment for constant G

% Antenna gain is function of frequency.  Apply gain vector here.
global Gain_Tx
Gain_Tx.signals.values = Gt_lin';
Gain_Tx.signals.dimensions = 1;
Gain_Tx.time = [];

global Gain_Rx
convert = sqrt(1e-6/T);%due to change in freq resolution in model
Gain_Rx.signals.values = Gt_lin'*convert;
Gain_Rx.signals.dimensions = 1;
Gain_Rx.time = [];

global LNAGain
freq_lna = load('LNAresponse.mat','X1');
Glna = load('LNAresponse.mat','Y1');
```

```
Glna_db = interp1(freq_lna.X1,Glna.Y1,(0:df:fs+3e9));
Glna_lin = sqrt(db2pow(Glna_db));
Glna_lin(isnan(Glna_lin)) = 0;
LNAGain.signals.values = Glna_lin';
LNAGain.signals.dimensions = 1;
LNAGain.time = [];

%% DETERMINE RADAR RANGE EQUATION PARAMETERS

%Signal Parameters
c = 299792458;                    %speed of light (m/s)
RefDelay = (1.7)*2/c;             %propagation delay of system
delay = 2*r/c - RefDelay;         %free space delay for each range
alpha = sqrt(1/((4*pi)^3*r^4)); %free space path loss
w = 2*0.3048;                     %width of flat target (2 ft) in m
h = w;                            %target is square
A = 4*pi*w^2*h^2;                 %numerator of RCS eq.
B = 400e6;                        %bandwidth
Grcs = sqrt(A*B*T);               %target gain factor

% Noise parameters
k = 1.38e-23;                     %Boltzmann's constant
T0 = 290;                         %standard temperature
F_db = 2.6+0.8+0.8;               %Noise Figs in dB (Filters/LNAs)
F = db2pow(F_db);                 %Noise Figure
system_noise = sqrt(k*T0*F*B);  %Additive system wgn per Hz

% LNA values (20 dB LNAs)
G_lna_db = 20;                    %Half gain in dB (voltage signal)
G_lna = sqrt(db2pow(G_lna_db)); %Linear gain of each LNA

% Noise Power: Data sheet has P=.01W, corresponding to -82 dBm/Hz
%   To get linear gain value applied to voltage signal and
%   corresponding to noise power, the Gain in dB/df is
%   -82 dBm/Hz - 30 + db_offset = -112 dB/Hz + db_offset =...
%   = P_dB in dB/df

P = 10^((-82-30+db_offset)/20);
% P = 0;                          %Transmitter Off

%% ESTIMATE ENVIRONMENTAL LOSSES
% Losses may be due to multiplath, antenna coupling, etc.
if horn == 0
    if strcmp(pol,'HH') == 1
        EnvLoss_db = 38;                    %Scaled for LPA HH pol
        EnvLoss = sqrt(db2pow(-EnvLoss_db));
    else
        EnvLoss_db = 23;                    %Scaled for LPA VV pol
        EnvLoss = sqrt(db2pow(-EnvLoss_db));
    end
else
    if strcmp(pol,'HH') == 1
        EnvLoss_db = 28;                    %Scaled for Horn HH pol
        EnvLoss = sqrt(db2pow(-EnvLoss_db));
    else
        EnvLoss_db = 31;                    %Scaled for Horn VV pol
        EnvLoss = sqrt(db2pow(-EnvLoss_db));
    end
end

%% SET SIMULINK PARAMETERS AND RUN THE SIMULATION

set_param([model,'/Noise Source/Noise Power'],...
    'Gain',num2str(P));                 %set noise signal variance
```

```
set_param([model,...
    '/Direct Conversion Receiver/Pulse Generator'],...
    'Period',num2str(1/rx_fs));         %set sample rate of ADC
set_param([model,'/System Noise/Noise Power'],...
    'Gain',num2str(system_noise*convert));  %set system noise var
set_param([model,'/LNA 1']...
    ,'Gain',num2str(G_lna));            %set gain value of LNA
set_param([model,'/LNA 2']...
    ,'Gain',num2str(G_lna));            %set gain value of LNA
set_param([model,'/Environment/Delay',num2str(1)],...
    'DelayTime',num2str(delay));           %set target range
set_param([model,'/Environment/PathLoss',num2str(1)],...
    'Gain',num2str(alpha*convert));        %set target path loss
set_param([model,'/Environment/RCS',num2str(1)],...
    'Gain',num2str(Grcs));                 %set target gain factor
set_param([model,'/Environment/EnvLoss'],...
    'Gain',num2str(EnvLoss));              %set target gain factor

tic
tout = sim(model, T, options, inputs);    %run simulation
toc

%% LOAD MEASURED SIGNALS (averages) OF AFIT RNR

% Load TX path data
filename = ['Splitter_100avg_unit',num2str(unit),'.xls'];
Yavg = xlsread(filename);                        %Values in dBm/MHz
Yavg = Yavg-10*log10(10^6);                      %Convert to dBm/Hz
Yavg_tx = Yavg;
favgmax = 2000;                          %Max freq of average (MHz)
favg = (0:favgmax/length(Yavg):favgmax-favgmax/length(Yavg));

```

```
filename = ['Source_100avg_unit',num2str(unit),'.xls'];
Yavg = xlsread(filename);                          %Values in dBm/MHz
Yavg = Yavg-10*log10(10^6);                        %Convert to dBm/Hz
Yavg_S = Yavg;

filename = ['BPF_100avg_unit',num2str(unit),'.xls'];
Yavg = xlsread(filename);                          %Values in dBm/MHz
Yavg = Yavg-10*log10(10^6);                        %Convert to dBm/Hz
Yavg_tx_bpf = Yavg;

%Load RX path data
if horn == 0
filename = ['2ftsq_',num2str(r),'m_',pol,'_rx_ant_100avg_',...
    num2str(unit),'_',num2str(time),'.xls'];
Yavg = xlsread(filename);                          %Values in dBm/df
Yavg = Yavg-db_offset;                             %Convert to dBm/Hz
Rx_ant = Yavg;

filename = ['2ftsq_',num2str(r),'m_',pol,'_rx_adc_100avg_',...
    num2str(unit),'_',num2str(time),'.xls'];
Yavg = xlsread(filename);                      %Values in dBm/df
Yavg = Yavg-db_offset;                             %Convert to dBm/Hz
Rx_adc = Yavg;
end

%Load Noise data
filename = 'System_noise.xls';
Yavg = xlsread(filename);                          %Values in dBm/df
Yavg = Yavg-db_offset;                             %Convert to dBm/Hz
Noise_sys = Yavg;
```

```
%% LOAD MEASURED SIGNALS (averages) USING STANDARD GAIN HORN

%Load RX path data
if horn == 1
filename = ['Horn_',num2str(r),'m_',pol,'_rx_ant_100avg_',...
    num2str(unit),'_',num2str(time),'.xls'];
Yavg = xlsread(filename);                           %Values in dBm/df
Yavg = Yavg-db_offset;                              %Convert to dBm/Hz
Horn_ant = Yavg;

filename = ['Horn_',num2str(r),'m_',pol,'_rx_adc_100avg_',...
    num2str(unit),'_',num2str(time),'.xls'];
Yavg = xlsread(filename);                           %Values in dBm/df
Yavg = Yavg-db_offset;                              %Convert to dBm/Hz
Horn_adc = Yavg;
end

%% LOAD SIMULATED SIGNAL AT DIFFERENT STAGES IN THE MODEL

LL = length(Ref.signals.values);
L1 = length(rx_env.signals.values);
freq = (0:rx_fs/LL:rx_fs-rx_fs/LL)/10^6;      %ADC freq vec in MHz
f = f./10^6;                 %frequency in MHz for plotting

% For the following signals: convert time domain voltage signals
%   to frequency domain signals in dBm/df then dBm/Hz
Noise = 20*log10(abs(fft(Noise.signals.values)))+30-10*log10(L);
Noise = Noise-db_offset;                            %Convert to dBm/Hz
N1 = 20*log10(abs(fft(Noise1.signals.values)))+30-10*log10(L);
N1 = N1-db_offset;                                  %Convert to dBm/Hz
N_bpf = ...
    20*log10(abs(fft(Noise_bpf.signals.values)))+30-10*log10(L);
```

```
N_bpf = N_bpf-db_offset;                            %Convert to dBm/Hz
TX_split = ...
    20*log10(abs(fft(tx_split.signals.values)))+30-10*log10(L);
TX_split = TX_split-db_offset;                      %Convert to dBm/Hz
TX = 20*log10(abs(fft(tx.signals.values)))+30-10*log10(L);
TX = TX-db_offset;                                  %Convert to dBm/Hz
RX_env = ...
    20*log10(abs(fft(rx_env.signals.values)))+30-10*log10(L1);
RX_env = RX_env-db_offset;                          %Convert to dBm/Hz
RX = 20*log10(abs(fft(rx.signals.values)))+30-10*log10(L1);
RX = RX-db_offset;                                  %Convert to dBm/Hz
RX_bpf = ...
    20*log10(abs(fft(rx_bpf.signals.values)))+30-10*log10(L1);
RX_bpf = RX_bpf-db_offset;                          %Convert to dBm/Hz
RX_adc = ...
    20*log10(abs(fft(rx_adc.signals.values)))+30-10*log10(L1);
RX_adc = RX_adc-db_offset;                          %Convert to dBm/Hz
SYS_NOISE = ...
    20*log10(abs(fft(sys_noise.signals.values)))+30-10*log10(L);
SYS_NOISE = SYS_NOISE-db_offset;                    %Convert to dBm/Hz
clear tx* rx*

half = floor(L/2);    %only need one side of fft spectrum to plot
half1 = floor(L1/2);
half2 = floor(LL/2);

%% DETERMINE SNR FROM CALCULATION, SIMULATION, AND MEASUREMENTS

% Calculate SNR as function of frequency and as integrated signal
fl = 400;                     %Low frequency of pass band in MHz
fh = 800;                     %High frequency of pass band in MHz
fl_ind = floor(400*T/1e-6);   %Low freq index
```

```
fh_ind = floor(800*T/1e-6);   %High freq index

%Simulated Signal
noise_sim = mean(SYS_NOISE(fl_ind:fh_ind))+pow2db(B); %Int noise
Pr_sim = mean(RX_adc(fl_ind:fh_ind))+pow2db(B); %Integrated rx
snr_sim = Pr_sim-noise_sim;          %Simulated SNR

%Measured Signal
if horn == 0
ind_l = ge(favg,400)';               %Get from 400 MHz to end
ind_h = le(favg,800)';               %Get from 0 to 800 MHz
Pr_pass = Rx_adc.*ind_l.*ind_h;
Pr_pass(Pr_pass == 0) = [];
Pr_meas = mean(Pr_pass)+pow2db(B);  %Integrated receive power
noise_pass = Noise_sys.*ind_l.*ind_h;
noise_pass(noise_pass == 0) = [];
noise_meas = mean(noise_pass)+pow2db(B);    %Integrated noise
% noise_meas = mean(Rx_adc(1:120)+db_offset);
snr_meas = Pr_meas-noise_meas;      %Measured SNR
end

%Measured Signal (Standard Gain Horn)
if horn == 1;
ind_l = ge(favg,400)';               %Get from 400 MHz to end
ind_h = le(favg,800)';               %Get from 0 to 800 MHz
Pr_pass = Horn_adc.*ind_l.*ind_h;
Pr_pass(Pr_pass == 0) = [];
Pr_meas = mean(Pr_pass)+pow2db(B);  %Integrated receive power
noise_pass = Noise_sys.*ind_l.*ind_h;
noise_pass(noise_pass == 0) = [];
noise_meas = mean(noise_pass)+pow2db(B);    %Integrated noise
% noise_meas = mean(Rx_adc(1:120)+db_offset);
```

```
snr_meas = Pr_meas-noise_meas;      %Measured SNR
end

%Calculated SNR
bpf_loss = 2;           %bandpass loss due to filter in dB
splitter_loss = 3.6;    %power loss due to power splitter in dB
Pt_calc_f = -82-30-bpf_loss-splitter_loss; %TX power at ant dB/Hz
Pt_calc = Pt_calc_f + pow2db(B);   %TX power in 400 MHz signal dB
if horn == 0
    G_calc = 6;             %constant antenna gain in dB for LNA
else
    G_calc = 12;            %constant gain for horn
end
G_calc_f = ones(1,401);
lambda_db_f = pow2db(c./((fl:fh)*10^6));%Vector of wavelengths dB
lambda_db = pow2db(c/600e6);    %center freq wavelength in dB
T_db = pow2db(T);               %measurement window in dB
rcs_db_f = pow2db(4*pi*w^2*h^2) - 2.*lambda_db_f;  %RCS vector dB
rcs_db = pow2db(4*pi*w^2*h^2) - 2*lambda_db;       %RCS scalar dB
snr_calc_f = Pt_calc_f + 2*G_calc_f + 2.*lambda_db_f +...
    rcs_db_f + pow2db(B) + T_db - (pow2db((4*pi)^3)...
    + pow2db(r^4) + pow2db(system_noise^2));        %in dB/Hz
snr_calc = Pt_calc + 2*G_calc + 2*lambda_db + rcs_db + ...
    pow2db(B) + T_db - (pow2db((4*pi)^3) + pow2db(r^4) + ...
    pow2db(system_noise^2));                        %in dB

%% CORRELATION PROCESSING (hardware max is T = 1 microsec)
if time == 1       %Don't perform correlation if T not 1 microsec

%% INTERPOLATION JUST LIKE THE A/D BOARD
interp_fs = 6e9;        %match to receiver interpolation (GSa/s)
interp_t = 0:1/(interp_fs):Ref.time(end);          %new time axis
```

```
interp_Rx = interp1(Rx.time,(Rx.signals.values),...
    interp_t,'spline');                            %interp Rx
interp_Ref = interp1(Ref.time,(Ref.signals.values),...
    interp_t,'spline');                            %interp Ref

%% SIMULINK MODEL CORRELATION PROCESSING
TxSig = interp_Ref;        %Reference signal
RxSig = interp_Rx;         %Receive signal
t = interp_t;              %time axis
XcorrLen = 5e-7;           %correlation window (integration time)
averages = floor(T/XcorrLen);  %# corrs to avg for each range est
averages = 1;                  %when comparing to measurements, 1

XcorrN = find(t>XcorrLen,1)-2;           %samples in correlation
num_xcorr = floor(t(end)/XcorrLen);      %# corr windows available
TxSig = TxSig( 1:XcorrN*num_xcorr );     %remove excess samples
RxSig = RxSig( 1:XcorrN*num_xcorr );     %remove excess samples
t = t(1:XcorrN*num_xcorr);               %remove excess samples
TxSig = reshape(TxSig,XcorrN,[])';       %reshape mx
RxSig = reshape(RxSig,XcorrN,[])';       %reshape mx
t = reshape(t,XcorrN,[])';               %reshape t matrix

Rxy = zeros(averages,XcorrN*2-1);    %setup Xcorr matrix
result.tar_loc = [];                 %setup target location array
result.tar_mag = [];                 %setup target magnitude array

rows = 1:averages;
%step through each Tx/Rx pair in the current block of signals
for j = 1:length(rows),
    Rxy(j,:) = abs(xcorr(RxSig(rows(j),:),TxSig(rows(j),:)));
end                                      %compute cross-corrlation
```

```
%find mean Xcorr for the current block of signals
meanRxy = abs(mean(Rxy(:,XcorrN:end),1));
meanRxy = meanRxy/1e-3;                  %convert to mW

%Determine the cross correlation with interpolated peaks
[¬,crosscorr,¬,¬] =peakfinderN_mod(meanRxy,1e-3,1,interp_fs*1e3);

dsample = c/(2*interp_fs);      %distance between each sample (m)
R = 0:c/(2*interp_fs):length(crosscorr)*dsample-dsample;  %vector

%% LOAD MEASURED CORRELATION RESULTS OF AFIT RNR

filename = ['unit',num2str(unit),'_',num2str(r),'m_',pol];
load(filename,'trace');
Corr_meas = trace;

filename = ['unit',num2str(unit),'_',pol,'_background'];
load(filename,'trace');
Corr_bg = trace;

R_meas = 0:c/(2*interp_fs):length(Corr_meas)*dsample-dsample;

%% LOAD MEASURED CORRELATION RESULTS (FROM STANDARD GAIN HORN)

filename = ['Horn_unit',num2str(unit),'_',num2str(r),'m_',pol];
load(filename,'trace');
Horn_Corr_meas = trace;

filename = ['Horn_unit',num2str(unit),'_',pol,'_background'];
load(filename,'trace');
Horn_Corr_bg = trace;
```

```
%% GET CORRELATION RESULTS (Simulated and Measured)

N = length(Ref.signals.values);     %Number of samples in signal

%Simulated peak-to-average sidelobe ratio
Skm2 = abs(xcorr(Rx.signals.values,Ref.signals.values)).^2;
Skm2 = Skm2(N:end);                     %Second half of xcorr
Corr_sim_db = pow2db(Skm2/max(Skm2));    %Normalized correlation
Avg_sidelobe_sim = mean(Corr_sim_db(1:600));
range_vec = Ref.time.*c./2;              %Range vector for plot
R_10 = R_meas(R_meas ≤ 10);          %Not concerned beyond 10 m
N_10 = length(R_10);                    %Number of samples to 10 m
crosscorr_db = pow2db(abs(crosscorr/max(crosscorr(1:N_10))).^2);

%Measured peak-to-average sidelobe ratio
Corr_db = pow2db(abs(Corr_meas/max(Corr_meas(1:N_10))).^2);
Corr_meas_db = pow2db(abs((Corr_meas-Corr_bg)/...
    max(Corr_meas(125:N_10)-Corr_bg(125:N_10))).^2);%Norm. corr.
Avg_sidelobe_meas = mean(Corr_meas_db(1:2700));     %Avg sidelobe

%Measured peak-to-average sidelobe ratio (Standard Gain Horn)
Horn_Corr_db = pow2db(abs(Horn_Corr_meas/...
    max(Horn_Corr_meas(1:N_10))).^2);
Horn_Corr_meas_db = pow2db(abs((Horn_Corr_meas-Horn_Corr_bg)/...
    max(Horn_Corr_meas(1:N_10)-...
    Horn_Corr_bg(1:N_10))).^2);           %Normalized correlation
Horn_Avg_sidelobe_meas = mean(Horn_Corr_meas_db(1:2700));

end
end
```

Bibliography

[1] *Electronic Warfare and Radar Systems Engineering Handbook*. Naval Air Warfare Center Weapons Division, 1999.

[2] Axelsson, S. R. J. "On the Theory of Noise Doppler Radar". *Geoscience and Remote Sensing Symposium, 2000. Proceedings. IGARSS 2000. IEEE 2000 International*, volume 2, 856–860 vol.2. 2000.

[3] Axelsson, S. R. J. "Noise Radar for Range/Doppler Processing and Digital Beamforming Using Low-bit ADC". *Geoscience and Remote Sensing, IEEE Transactions on*, 41(12):2703–2720, 2003.

[4] Axelsson, S. R. J. "Noise Radar Using Random Phase and Frequency Modulation". *Geoscience and Remote Sensing, IEEE Transactions on*, 42(11):2370–2384, 2004.

[5] Axelsson, Sune R. J. "Generalized Ambiguity Functions for Ultra Wide Band Random Waveforms". *Radar Symposium, 2006. IRS 2006. International*, 1–4. 2006.

[6] Bell, D. C. and R. M. Narayanan. "ISAR Turntable Experiments Using a Coherent Ultra Wide-band Random Noise Radar". 1764–1767. July 1999 1999.

[7] Dawood, M. and R. M. Narayanan. "Ambiguity Function of an Ultrawideband Random Noise Radar". *Antennas and Propagation Society International Symposium, 2000. IEEE*, volume 4, 2142–2145 vol.4. 2000.

[8] Dawood, M. and R. M. Narayanan. "Generalised wideband ambiguity function of a coherent ultrawideband random noise radar". *Radar, Sonar and Navigation, IEE Proceedings* -, 150(5):379–386, 2003. ID: 1.

[9] Garmatyuk, D. S. and R. M. Narayanan. "SAR Imaging Using a Coherent Ultrawideband Random Noise Radar". William Miceli I. (editor), *SPIE*, volume 3810, 223–230. 1999.

[10] Guosui, Liu, Gu Hong, and Su Weimin. "Development of Random Signal Radars". *Aerospace and Electronic Systems, IEEE Transactions on*, 35(3):770–777, 1999.

[11] Kay, Steven. *Intuitive Probability and Random Processes using MATLAB*. Springer, New York, NY, 2006. ISBN 978-0387241579.

[12] Kulpa, K., K. Lukin, W. Miceli, and T. Thayaparan. "Signal Processing in Noise Radar Technology [Editorial]". *Radar, Sonar Navigation, IET*, 2(4):229–232, 2008.

[13] Kulpa, Krzysztof. *Continuous Wave Radars, Monostatic, Multistatic and Network*, volume 2 of *Advances in Sensing with Security Applications*, 215–242. Springer Netherlands, 2006.

[14] Kwag, Y. K. and C. H. Chung. "UAV Based Collision Avoidance Radar Sensor". *Geoscience and Remote Sensing Symposium, 2007. IGARSS 2007. IEEE International*, 639–642. 2007.

[15] Lai, C. P., R. M. Narayanan, and G. Culkowski. "Through Wall Surveillance Using Ultrawideband Random Noise Radar". *25th Army Science Conference*, 1–4. Nov. 2006.

[16] Levanon, N. and E. Mozeson. *Radar Signals*. Wiley-IEEE Press, 2004. ISBN 978-0471473787.

[17] Li, Zhixi and R. M. Narayanan. "Doppler Visibility of Coherent Ultrawideband Random Noise Radar Systems". *Aerospace and Electronic Systems, IEEE Transactions on*, 42(3):904–916, 2006.

[18] Lievsay, James R. "Simultaneous Range/Velocity Detection with an Ultrawideband Random Noise Radar Through Fully Digital Cross-correlation in the Time Domain", Master's Thesis, Air Force Institute of Technology, 2011.

[19] Lievsay, James R. and Geoffrey A. Akers. "Moving Target Detection Via Digital Time Domain". 2011.

[20] Ludwig, Matt T. "UHF Antenna Redesign for AFIT Random Noise Radar", Master's Thesis, Air Force Institute of Technology, 2012.

[21] Lukin, K. A., A. A. Mogyla, Yu A. Alexandrov, O. V. Zemlyaniy, T. Lukina, and Yu Shiyan. "W-band Noise Radar Sensor for Car Collision Warning Systems". *Physics and Engineering of Millimeter and Sub-Millimeter Waves, 2001. The Fourth International Kharkov Symposium on*, volume 2, 870–872 vol.2. 2001.

[22] Lukin, K. A. and R. M. Narayanan. "Fifty Years of Noise Radar". *Physics and Engineering of Microwaves, Millimeter and Submillimeter Waves (MSMW), 2010 International Kharkov Symposium on*, 1–3. 2010.

[23] Lukin, Konstantin A. "Noise Radar with Correlation Receiver as the Basis of Car Collision Avoidance System". *Microwave Conference, 1995. 25th European*, volume 1, 506–507. 1995.

[24] Meller, M. "Some Aspects of Designing Real-time Digital Correlators for Noise Radars". *Radar Conference, 2010 IEEE*, 821–825. 2010. ISBN 1097-5659.

[25] Narayanan, R. M. and M. Dawood. "Doppler Estimation Using a Coherent Ultrawide-band Random Noise Radar". *Antennas and Propagation, IEEE Transactions on*, 48(6):868–878, 2000.

[26] Narayanan, Ram M., Yi Xu, Paul D. Hoffmeyer, and John O. Curtis. "Design, Performance, and Applications of a Coherent Ultra-wideband Random Noise Radar". *Optical Engineering*, 37(6):1855–1869, June 1998 1998. URL `http://link.aip.org/link/?JOE/37/1855/1`.

[27] Nelms, Matthew E. "Development and Evaluation of a Multistatic Ultrawideband Random Noise Radar", Master's Thesis, Air Force Institute of Technology, 2010.

[28] Oppenheim, A. and R. Schafer. *Discrete-time Signal Processing.* Prentice Hall, Upper Saddle River, NJ, 3rd edition, 2009. ISBN 978-0131988422.

[29] Pace, Philip E. *Detecting and Classifying Low Probability of Intercept Radar, Second Edition.* Artech House, Boston, MA, 2 edition, 2009. ISBN 978-1-59693-234-0.

[30] Priestly, John A. "AFIT NoNET Enhancements: Software Model Development and Optimization of Signal Processing Architecture", Master's Thesis, Air Force Institute of Technology, 2011.

[31] Richards, Mark A., James A. Scheer, and William A. Holm. *Principles of Modern Radar: Basic Principles.* Scitech Publishing, Inc., Raleigh, NC, 2010. ISBN 9781891121524.

[32] Rihaczek, August W. "Delay-Doppler Ambiguity Function for Wideband Signals". *Aerospace and Electronic Systems, IEEE Transactions on*, AES-3(4):705–711, 1967.

[33] Schmitt, A. "Radar Imaging with a Network of Digital Noise Radar Systems", Master's Thesis, Air Force Institute of Technology, 2009.

[34] Schmitt, A. and P. Collins. "Demonstration of a Network of Simultaneously Operating Digital Noise Radars [Measurements Corner]". *Antennas and Propagation Magazine, IEEE*, 51(2):125–130, 2009.

[35] Taylor, James D. *Ultra-wideband Radar Technology.* CRC Press, Boca Raton, Florida, 2001. ISBN 978-0-8493-4267-7.

[36] Thayaparan, T., M. Dakovic, and L. Stankovic. "Mutual Interference and Low Probability of Interception Capabilities of Noise Radar". *Radar, Sonar Navigation, IET*, 2(4):294–305, 2008.

[37] Thayaparan, T. and C. Wernik. *Noise Radar Technology Basics.* Technical Report TM 2006-266, Defence Research and Devlopment Canada - Ottawa, 2006.

[38] Theron, I. P., E. K. Walton, S. Gunawan, and Lixin Cai. "Ultrawide-band Noise Radar in the VHF/UHF Band". *Antennas and Propagation, IEEE Transactions on*, 47(6):1080–1084, 1999.

[39] Walden, R. H. “Analog-to-Digital Converter Survey and Analysis”. *Selected Areas in Communications, IEEE Journal on*, 17(4):539–550, 1999.

[40] Weiss, L. G. “Wavelets and Wideband Correlation Processing”. *Signal Processing Magazine, IEEE*, 11(1):13–32, 1994.

[41] Woodward, Philip M. *Probability and Information Theory, with Applications to Radar*. McGraw-Hill, New York, 1953. ISBN 0890061033.

[42] Xu, X. and R. M. Narayanan. “FOPEN SAR Imaging UWB Step-frequency and Random Noise Waveforms”. *IEEE Transactions on Aerospace and Electronic Systems*, 37(4):1287–1300, 2001.

[43] Xu, Yi, R. M. Narayanan, X. Xu, and J. O. Curtis. “Polarimetric Processing of Coherent Random Noise Radar Data for Buried Object Detection”. *IEEE Transactions on Geosciences and Remote Sensing*, 39(3):467–478, Mar. 2001.

REPORT DOCUMENTATION PAGE	*Form Approved OMB No. 0704–0188*

1. REPORT DATE *(DD–MM–YYYY)* 22–03–2012	**2. REPORT TYPE** Master's Thesis	**3. DATES COVERED** *(From — To)* Aug 2010 — Mar 2012

4. TITLE AND SUBTITLE Simultaneous Range-Velocity Processing and SNR Analysis of AFIT's Random Noise Radar	**5a. CONTRACT NUMBER**
	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S) Thorson, Timothy J, Capt, USAF	**5d. PROJECT NUMBER**
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Air Force Institute of Technology Graduate School of Engineering and Management (AFIT/EN) 2950 Hobson Way WPAFB OH 45433-7765	**8. PERFORMING ORGANIZATION REPORT NUMBER** AFIT/GE/ENG/12-40
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) Intentionally Left Blank	**10. SPONSOR/MONITOR'S ACRONYM(S)**
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

13. SUPPLEMENTARY NOTES

This material is declared a work of the U.S. Government and is not subject to copyright protection in the United States.

14. ABSTRACT

This paper presents two research objectives aimed at advancing the AFIT RNR signal processing algorithm and modeling capability toward the overarching goal of performing collision avoidance on an autonomous vehicle. In both research efforts, analytical, simulated, and measured results are provided and used to draw research conclusions. The first research effort is aimed at reducing the memory required for 2D processing in the time domain in order to distribute the processing algorithm across hundreds of processors on a GPU. Distributed processing reduces the overall 2D processing time and the feasibility of a near real-time implementation is studied. The second effort consists of improving a Simulink® model of the AFIT RNR. Each component of the AFIT RNR, as well as the target environment, is modeled and compared to measured results. A robust model will provide a useful tool to study the signal-to-noise ratio (SNR) of the RNR at all points within the radar system.

15. SUBJECT TERMS

Noise Radar, Collision Avoidance, Autonomous Navigation, Low Probability of Intercept Radar

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON** Geoffrey A. Akers, Lt Col, USAF (ENG)
a. REPORT	**b. ABSTRACT**	**c. THIS PAGE**			**19b. TELEPHONE NUMBER** *(include area code)*
U	U	U	UU	156	(937) 255-3636 x4659; geoffrey.akers@afit.edu

Standard Form 298 (Rev. 8–98)
Prescribed by ANSI Std. Z39.18